Intégration de Publish/Subscribe avec Top-k dans le système Real-Time Bidding

par

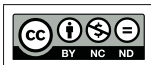
Sonia Slimani

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE LOGICIEL
M. Sc. A.

MONTRÉAL, LE "02 JUIN 2020"

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

Sonia Slimani, 2020



Cette licence Creative Commons signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette oeuvre à condition de mentionner l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'oeuvre n'ait pas été modifié.

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Kaiwen Zhang, directeur de mémoire
Département de génie logiciel et des TI, École de technologie supérieure

M. Pascal Giard, président du jury
Département de génie électrique, École de technologie supérieure

M. Julien Gascon-Samson, membre du jury
Département de génie logiciel et des TI, École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE "22 MAI 2020"

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier mon directeur de mémoire M. Kaiwen Zhang pour sa disponibilité, son encouragement et ses conseils judicieux tout au long de mon projet de recherche.

J'adresse un grand merci à mes parents et mes frères pour leur soutien moral et leurs encouragements durant ma maitrise.

Enfin, j'adresse mes sincères remerciements à mes amis et les membres de "Fusée Lab" de m'avoir soutenu et encouragé pour l'accomplissement de ce mémoire.

Intégration de Publish/Subscribe avec Top-k dans le système Real-Time Bidding

Sonia Slimani

RÉSUMÉ

Le système d'enchère en temps réel ou Real-Time Bidding (RTB) a connu récemment une croissance massive dans le marché du marketing en ligne. Les technologies RTB permettent à Ad Exchange (AdX) de mener des enchères en ligne afin de vendre des espaces publicitaires ciblés en sollicitant des offres auprès d'acheteurs potentiels, appelés Demand-Side Platforms (DSPs).

Dans OpenRTB, une spécification des protocoles et normes bien connue de RTB, AdX diffuse les demandes d'offres à tous les DSPs pour chaque enchère. Ce protocole de communication n'est pas très efficace, car pour chaque enchère seulement un nombre limité de DSPs soumet réellement des réponses non vides et compétitives à AdX. L'échange de requêtes d'enchères non pertinentes et des offres non compétitives gaspille des ressources de calcul et de communication importantes.

Dans ce mémoire, nous proposons d'intégrer un modèle Publish/Subscribe (Pub/Sub) avec filtrage Top-k dans RTB. En particulier, nous utilisons un système Pub/Sub basé sur le contenu afin d'exprimer les intérêts des DSPs, ce qui permet une diffusion sélective des requêtes d'enchères pour éliminer les réponses vides.

Nous formulons également le problème d'optimisation du nombre de réponses aux enchères et nous proposons de combiner le filtrage Top-k avec l'analyse de régression des variables continues comme solution heuristique pour réduire davantage le nombre de réponses non compétitives. Nous adoptons également des modèles discrets pour une exécution plus rapide.

Enfin, nous évaluons nos deux solutions proposées par rapport à l'approche de base OpenRTB en termes de temps total de traitement, prix payé à la fin d'enchère et efficacité.

Mots-clés: Real time bidding, publicité en ligne, Publish/Subscribe basé sur le contenu, filtrage Top-k, apprentissage automatique

Integration of Publish/Subscribe with Top-k filtering in Real-Time Bidding

Sonia Slimani

ABSTRACT

Real-Time Bidding (RTB) advertising has recently experienced a massive growth in the industry of online marketing. RTB technologies allow an Ad Exchange (AdX) to conduct online auctions in order to sell targeted ad impressions by soliciting bids from potential buyers, called Demand-Side Platforms (DSPs).

In the OpenRTB specifications, which is a well-known open standard protocol for RTB, the AdX sends bid requests to all DSPs for every auction. This communication protocol is highly inefficient since for each given auction, only a small fraction of DSPs will actually submit a competitive bid to the AdX. The exchange of bid requests to uninterested parties waste valuable computation and communication resources.

In this thesis, we propose to leverage publish/subscribe to optimize the auction protocol used in RTB. In particular, we demonstrate how RTB semantics can be expressed using content-based subscriptions, which allows for selective dissemination of bid requests in order to eliminate no-bid responses.

We also formulate the problem of minimizing the number of bid responses per auction, and propose combining Top-k scoring with regression analysis with continuous variables as a heuristic solution to further reduce the number of irrelevant responses. We then adapt our solution by considering discrete machine learning models for faster execution.

Finally, we evaluate our proposed solutions against the OpenRTB baseline implementation in terms of end-to-end latency and total paid price over time efficiency.

Keywords: Real time bidding, online advertising, content-based Publish/Subscribe system, Top-k filtering, machine learning

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

RTB	Real-Time Bidding
AdX	Ad Exchange
DSP	Demand-Side Platform
Pub/Sub	Publish/Subscribe

INTRODUCTION

Real-Time Bidding (RTB) est une innovation transformatrice dans le marché de publicité en ligne, il a permis de passer de la vente des publicités indéterminée à grande échelle, en vente personnalisée. Il aide les éditeurs des pages web à vendre en temps réel des espaces publicitaires au plus offrant grâce aux enchères. L'annonceur choisit d'enchérir ou non et à quel prix selon certaines caractéristiques tels que la taille de la bannière, le contexte de la page web, le profil utilisateur, etc.

IAB (2016) a proposé une spécification du système RTB appelée OpenRTB, qui fournit des normes industrielles pour la communication entre acheteurs et vendeurs d'espaces publicitaires. Elle est basée sur deux composants principaux : Ad Exchange (AdX) et Demand-Side Platform (DSP). AdX, étant un agent intermédiaire entre les vendeurs et les acheteurs d'espaces publicitaires, génère une requête pour chaque enchère et l'envoie à tous les DSPs éligibles. D'un autre côté, l'annonceur demande au DSP de lancer une campagne publicitaire en fonction d'un produit particulier, un public cible et un budget prédéfini pendant une durée spécifique selon Mullarkey & Hevner (2015). Dès la réception de requête d'enchère de l'AdX, chaque DSP calcule le prix d'offre à proposer si l'espace d'annonce intéresse l'un de ses clients. Ensuite, AdX collecte les offres et lance une enchère pour choisir le DSP proposant le meilleur prix pour afficher son annonce comme dans la figure 0.1. Le gagnant paie le prix lié au mécanisme d'enchères utilisé, le mécanisme le plus utilisé actuellement est l'enchère de Vickrey d'après Yuan, Wang, Li & Qin (2014).

Dans la spécification OpenRTB, le DSP ne participe qu'aux enchères qui intéressent ses annonceurs en proposant un prix d'offre, sinon il renvoie une réponse vide. Les requêtes et les réponses aux enchères sont échangées en utilisant le protocole HTTP, au format JSON selon IAB (2016), qui est un format lourd et complexe, car il est lisible par l'humain. Ainsi, un délai d'expiration qui est fixé pour chaque enchère (appelé t_{max}), est défini pour collecter

Figure 0.1 Page web contenant des espaces publicitaires avec des annonces

toutes les réponses aux enchères, y compris les réponses vides. Par conséquent, l'échange des requêtes d'enchères non pertinentes et des réponses vides, ainsi que les réponses aux enchères avec des prix non compétitifs qui n'ont aucune chance réaliste de gagner une enchère, gaspille des ressources importantes du système. Donc, nous avons besoin d'une solution permettant de sélectionner un sous-ensemble de DSPs à contacter pour chaque enchère afin de réduire les surcouts de calcul et de communication dans le système RTB.

Dans ce mémoire, nous proposons d'intégrer le modèle Publish/Subscribe (Pub/Sub) dans le système RTB pour permettre aux DSPs d'exprimer à AdX les intérêts de leurs annonceurs sous forme d'abonnements basés sur le contenu (Eugster, Felber, Guerraoui & Kermarrec (2003)). Par la suite, AdX filtre les espaces d'annonces et n'envoie que ceux qui correspondent aux abonnements des DSPs. Cela pourrait diminuer le nombre des messages échangés et le temps de traitement total.

De plus, nous pouvons réduire les frais généraux en éliminant les offres non concurrentielles pour chaque enchère. Nous ajoutons donc à notre système Pub/Sub un filtrage Top-k avec des techniques d'apprentissage automatique pour sélectionner dynamiquement un sous-ensemble de DSPs susceptibles de soumettre des offres compétitives pour une enchère donnée.

Dans ce mémoire, nous fournissons les contributions suivantes :

- intégrer le modèle Pub/Sub dans RTB, modéliser la requête d'enchère en tant que publication et exprimer les intérêts de DSP en tant qu'abonnement basé sur le contenu (section 3.1),
- formuler le problème d'optimisation du nombre de réponses aux enchères RTB (section 3.2),
- utiliser le filtrage Top-k pour sélectionner les meilleurs DSPs l'aide des techniques d'apprentissage automatique (section 3.3),
- implémenter un générateur de requêtes d'enchères ajustables issu de la base de données iPinYou en utilisant les mêmes distributions de ses attributs (Section 4.3), et
- mettre en œuvre nos approches en utilisant Header Exchange de RabbitMQ (section 4.1), les évaluer en utilisant la base de données générée et les comparer par rapport à l'approche de base OpenRTB. (chapitre 6)

Nous définissons aussi certaines notions de base relatives à notre travail dans le chapitre 1. Ensuite, nous présentons dans ce mémoire une revue de littérature dans le chapitre 2. Puis, nous parlons de notre expérience personnelle durant notre recherche dans le chapitre 5. Enfin, nous concluons dans la conclusion.

CHAPITRE 1

NOTIONS DE BASE

Afin de mieux comprendre notre projet de recherche, nous définissons quelques notions de base qui sont Real-Time bidding, l'enchère de Vickrey, la spécification OpenRTB, la base de données iPinYou, Publish/Subscribe et les techniques d'apprentissage automatique.

1.1 Real-Time Bidding

Real-Time Bidding ou enchère en temps réel (RTB) est une forme de publicité en ligne qui fournit l'achat et la vente d'espaces publicitaires en ligne à un public cible, via des enchères en temps réel qui se produisent pendant le temps de chargement d'une page web (p. ex., entre 200 ms et 300 ms Kumar (2017)).

Comme le montre la figure 1.1, le processus typique de RTB selon Yuan *et al.* (2014) démarre lorsque l'annonceur demande à un DSP de lancer une campagne publicitaire pour un produit particulier en fonction d'un budget prédéfini et internaute cible pendant une durée spécifique.

1) Une fois qu'un utilisateur visite une page Web, (2) AdX génère une requête d'enchère pour chaque requête d'annonce contenant les caractéristiques d'espace publicitaire, d'utilisateur et d'enchère (3) et l'envoie à tous les DSPs éligibles. (4) Après, chacun décide de participer ou non dans l'enchère en comparant l'espace d'annonce ainsi que les informations détaillées d'internaute fournies par Data Management Platform (DMP) par rapport aux intérêts de ses clients. Si oui, il calcule le prix d'offre et l'envoie dans la réponse à l'enchère, sinon il renvoie une réponse vide. Ensuite, AdX collecte les offres avant t_{max} : les offres reçues après t_{max} sont ignorés par AdX. (5) Enfin, AdX lance une enchère pour choisir le DSP avec le meilleur prix pour (6) afficher son annonce dans la page web du Publisher (7) visualisé par l'utilisateur. Le gagnant paie le deuxième meilleur prix puisque RTB utilise généralement l'enchère de Vickrey.

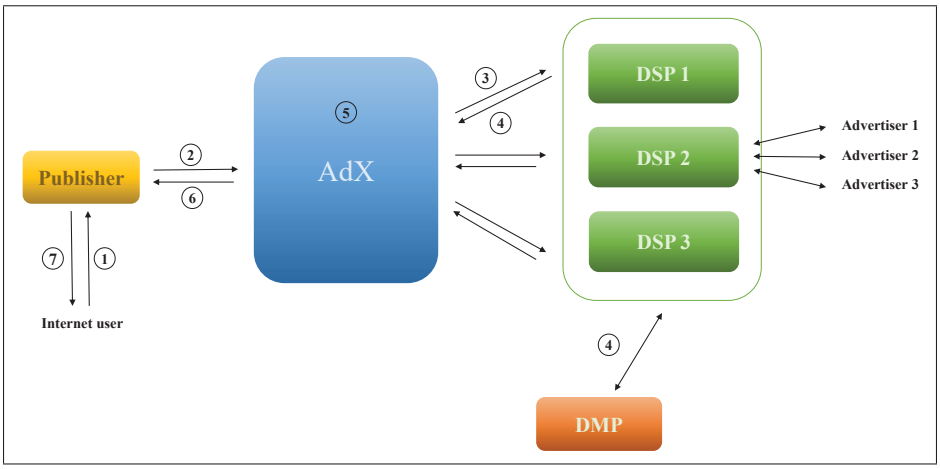


Figure 1.1 Présentation du processus RTB

Dans notre travail, nous nous concentrons sur l'étape (3) en modifiant le comportement d'AdX. Au lieu de diffuser la requête d'enchère à tous les DSPs, il la filtre par rapport aux intérêts des DSPs, en utilisant Pub/Sub, pour décider ceux qui recevront la demande d'offre.

1.2 L'enchère de Vickrey

L'enchère de Vickrey est un type d'enchère où les participants soumettent leurs offres pour un produit sans connaître les prix proposés par les autres. Ensuite, le meilleur offrant gagne mais paie le deuxième meilleur prix, ce qui encourage tout le monde à enchérir honnêtement. Selon Kalra, Borcea, Wang & Chen (2019), dans une vente aux enchères de Vickrey le responsable fixe dès le début un prix de réserve, qui est un prix minimum au-dessous duquel le produit ne peut pas être vendu. Dans ce cas, le prix à payer est calculé selon le théorème 1.

Theorem 1. *Soit* $\{b_1 ....b_n\}$ *un ensemble des participants à une enchère et R le prix de réservation :*

- *si* $\exists\, b_i > \max_{j \neq i} b_j \Rightarrow b_i$ *est le meilleur offrant et paie le deuxième meilleur prix.*

- *si* $b_i = \max_{j \neq i} b_j$ *(c'est-à-dire il y a deux offrants de meilleur prix), le gagnant de l'enchère est choisi arbitrairement ou selon son ordre alphabétique, et paie son prix d'enchère.*

- *si* $max_{j \neq i} b_j \leq R$*, le gagnant paie R.*

1.3 La spécification OpenRTB

OpenRTB est un projet développé par le laboratoire IAB Technology pour spécifier des protocoles et des normes de communication entre acheteurs et vendeurs d'impressions publicitaires dans le contexte de la publicité RTB dans IAB (2016). Il propose une API de RTB avec tous les objets essentiels (AdX et DSP) et les interactions entre eux (requête d'enchère, réponse à l'enchère, facturation de gain et avis de perte) comme montre la figure 1.2. Le RTB Exchange (AdX) envoie une requête d'enchère au Bidder (DSP) avec des détails sur le site, le contenu, l'utilisateur, l'appareil, l'emplacement, etc. (tableau 1.1), ensuite le DSP retourne la réponse à l'enchère (tableau 1.2) avec un prix d'offre s'il décide de participer à l'enchère ou réponse vide sinon. Un DSP retourne une réponse vide à cause des différentes raisons citées dans le tableau 1.3. Après le lancement de l'enchère, AdX envoie un avis de gain au gagnant et un avis de perte aux autres DSPs.

Dans ce mémoire, nous nous concentrons sur les raisons possibles des réponses vides. Nous analysons les différentes raisons et nous identifions celles qui pourraient être évitées par l'intégration d'un système de Pub/Sub.

1.4 La base de données iPinYou

La base de données iPinYou est un ensemble de journaux d'enchères, d'impressions, de clics et de conversion provenant de certaines campagnes publicitaires réelles exécutés via la plateforme DSP de iPinYou pour une certaine période afin d'évaluer les algorithmes d'enchères des DSPs soumis

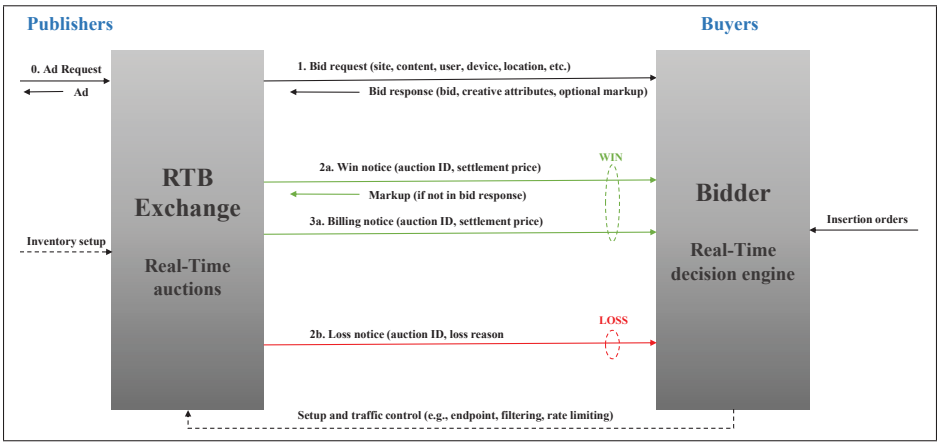


Figure 1.2 Le processus de RTB selon la spécification OpenRTB

Tableau 1.1 Requête d'enchère (étape 2)

Attribut	Type	Description
id	String ; obligatoire	identifiant unique de requête d'enchère
imp	Object array ; obligatoire	tableau des espaces publicitaires offerts
site	Object ; recommandé pour les sites web	détails à propos de l'éditeur du site web
app	Object ; recommandé pour applications	détails à propos de l'éditeur de l'application
device	Object ; recommandé	détails à propos de l'utilisateur de l'appareil
user	Object ; recommandé	détails à propos du public cible
test	Integer ; par défaut 0	indicateur du mode dans lequel les enchères ne sont pas facturables (0 :live mode, 1 :test mode)
at	Integer ; par défaut 2	type d'enchère : 1= meilleur prix, 2= deuxième meilleur prix
t_{max}	Integer	timeout in ms
wseat	String array	liste des acheteurs autorisés à enchérir sur cette espace

lors d'une compétition en 2013 (Liao, Peng, Liu & Shen (2014)). Selon notre connaissance, il s'agit de la seule base de données accessible au public contenant des enchères RTB.

Tableau 1.2 Réponse à l'enchère (étape 3)

Attribut	Type	Description
id	String ; obligatoire	identifiant unique de la réponse à l'enchère
seatbid	Object array	tableau de seatbid objects
site	Object ; recommandé pour les sites web	détails à propos de l'éditeur du site web
bidid	String	identifiant de réponse générée par le soumissionnaire
cur	String ; par défaut "USD"	les enchères utilisent actuellement alpha codes ISO4217
customdata	String	fonction facultative permettant à un soumissionnaire d'envoyer des données dans le cookie de l'AdX
nbr	Integer	raison pour ne pas participer à l'enchère
ext	Object	Espace réservé pour le soumissionnaire

Tableau 1.3 Les causes de réponse vide avec leurs codes

Code	Cause de réponse vide
0	Erreur inconnu
1	Erreur technique
2	Erreur invalide
3	Web spider
4	Trafic non humain suspect
5	Cloud, data center ou proxy IP
6	Appareil non supporté
7	Publisher ou site bloqué
8	Profil utilisateur non intéressant
9	Plafond de lecteur quotidien atteint
10	Plafond de domaine quotidien atteint

Dans ce mémoire, nous nous intéressons uniquement aux journaux d'enchères, qui contiennent des informations sur les requêtes d'enchères et les réponses des DSPs. Le tableau 1.4 résume les différents attributs trouvés dans la base de données iPinYou, qui correspondent parfaitement aux spécifications d'OpenRTB. Nous utilisons cette base de données dans la section 4.3 afin

d'implémenter un générateur de requêtes d'enchères ajustable à utiliser dans l'évaluation (chapitre 6).

Tableau 1.4 Les attributs de la base de données iPinYou

Attribut	**Description**
BidID	identifie une impression d'annonce. Il s'agit d'une chaîne aléatoire [32] de (a..z) et (0..9).
Timestamp	indique quand la requête d'enchère arrive au DSP.
Type de journal	1 (impression), 2 (clic) ou 3 (conversion).
iPinYou ID	est un ID haché de cookie utilisateur défini par iPinYou.
User-Agent	est le navigateur utilisé par l'utilisateur pour visiter le site Web des éditeurs.
Adresse IP	est l'adresse IP de l'utilisateur sans le dernier octet afin de protéger la confidentialité de l'utilisateur.
ID de région et ID de ville	sont la région et la ville où l'utilisateur se connecte.
Ad exchange	représente de quel AdX provient cette annonce, il peut être 1, 2, 3, 4, 5 ou 6, qui représente Tanx (Alibaba), Adx (Google DoubleClick AdX), Tencent (Tencent), Baidu (Baidu), Youku (Youku) ou Amx (Google Mobile) respectivement.
ID de l'espace publicitaire	indique l'emplacement où l'impression d'annonce apparaîtra sur la page Web, et Largeur et hauteur de l'espace publicitaire sont respectivement la largeur et la hauteur de l'espace publicitaire.
AdSlotFormat	est un espace publicitaire (1), une fenêtre contextuelle (2) ou inconnu (0).
AdSlotFloorPrice	est le prix le plus bas auquel l'éditeur autorise les DSP à gagner l'espace publicitaire. Si aucun DSP enchère n'est plus haut que le prix plancher, cette enchère n'a pas de DSP gagnant, iPinYou fait une échelle linéaire de prix d'enchère avant la publication du journal.
AdvertiserID	représente les annonceurs.
UserProfileIDs	identifiants des noms de catégorie DAAT (Digital Audience Advertising Taxonomy). iPinYou développe DAAT pour décrire les informations utilisateur à partir de perspectives démographiques, géographiques, d'intérêt à long terme et d'achat sur le marché.

1.5 Publish/Subscribe

Le système Publish/subscribe (Pub/Sub) est un paradigme de communication permettant aux éditeurs de diffuser des publications en temps réel à leurs abonnés intéressés.

Comme montre la figure 1.3, (1) chaque abonné peut exprimer son intérêt pour une publication particulière en envoyant un abonnement au service intermédiaire de Pub/Sub. (2) Une fois que l'éditeur diffuse cette publication, (3) il la livre à ses abonnés correspondants.

Selon Eugster *et al.* (2003), le système Pub/Sub est caractérisé par la scalabilité, une topologie dynamique et le découplage en termes de :

- espace : les éditeurs et les abonnés ne se connaissent pas et ne savent ni qui envoie ni qui reçoit ni même combien d'entités participant à l'interaction
- temps : pas besoin d'interaction en même temps
- synchronisation : il est asynchrone

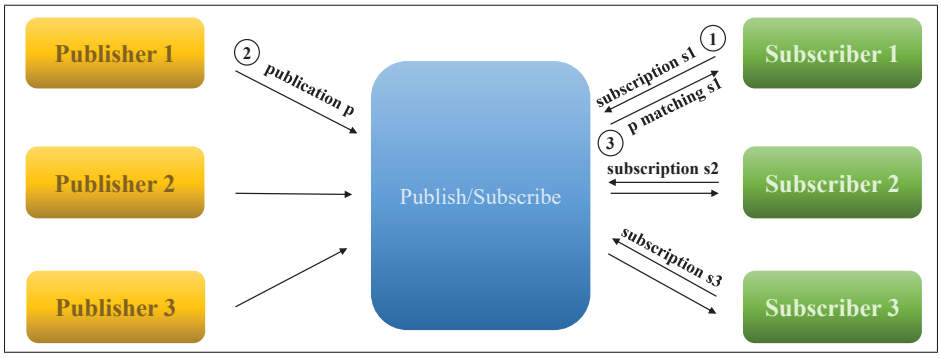


Figure 1.3 Présentation du système Publish/Subscribe

1.5.1 Implémentations

1.5.1.1 Kafka

Kafka est un système Pub/Sub développé et utilisé par LinkedIn pour collecter et distribuer de gros volumes d'évènements et de données des journaux avec une faible latence. Il intègre des agrégateurs de journaux et des systèmes de messagerie existantes et il fonctionne en ligne et hors ligne (Thein (2014)). Dans Kafka, les producteurs publient des messages identifiés par des sujets et les consommateurs s'abonnent à ces sujets pour recevoir les messages désirés. Chaque sujet est divisé en plusieurs partitions et chaque broker dans le réseau de kafka stocke une ou plusieurs de ces partitions afin d'équilibrer la charge. La coordination au sein de ce système est faite à l'aide d'un coordinateur appelé Zookeeper comme montre la figure 1.4.

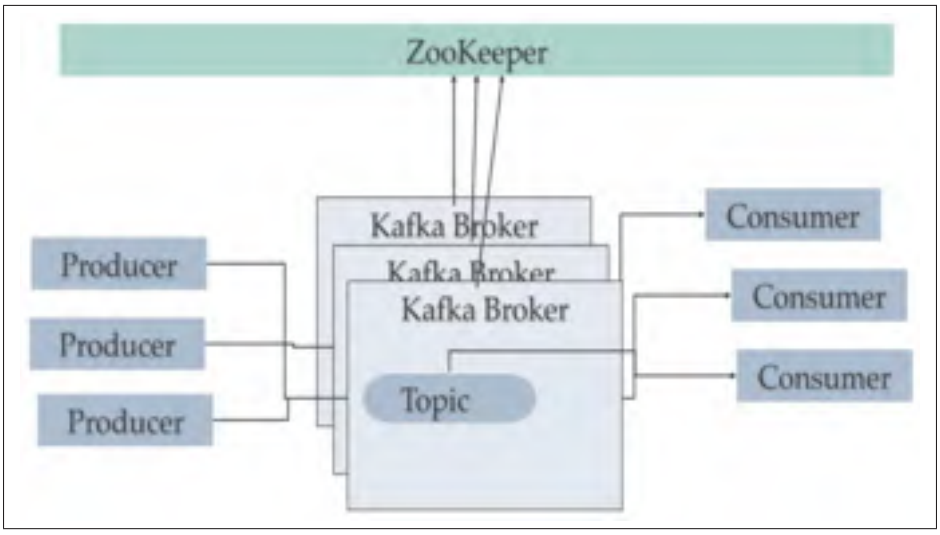


Figure 1.4 L'architecture de Kafka, référence : Cloudurable (2017)

1.5.1.2 RabbitMQ

RabbitMQ est un système Pub/Sub sorti en 2007 et développé en Erlang selon Ionescu (2015). Dans RabbitMQ, le producteur envoie le message au service d'échange, qui le place dans une file d'attente jusqu'à sa récupération de ses destinataires.

Donc sa tâche la plus importante est de déterminer dans quelles et combien de files d'attente le message doit être acheminé. Les messages peuvent être distribués selon quatre types de base :

- **Direct Exchange :** le producteur ajoute un routing key au message et l'envoie au service d'échange qui cherche la file d'attente qui correspond à cette clé. Une fois la correspondance est trouvée, l'achemine vers cette file d'attente. Donc il s'agit d'une communication directe entre émetteur et destinataire (figure 1.5).
- **Topic Exchange :** dans ce cas, les abonnées sont liés à des files d'attente caractérisées par des sujets spécifiques. Une fois le producteur publie un message, le service d'échange l'achemine vers les files correspondantes selon son sujet. Il est caractérisé par le routage basé sur hiérarchie, c'est à dire chaque consommateur peut s'abonner à une hiérarchie de sujets représentée par une liste de sujets séparés par ".", par exemple "topic1.topic11.topic111". La liste des sujets peut contenir aussi "*" pour remplacer la position d'un mot, par exemple "topic1.*.topic111" correspond au sujet "topic111" appartenant au premier sujet "topic1". Un symbole "#" indique zéro ou plusieurs mots (par exemple, "topic1.topic11.#" correspond à tous les sujets sous "topic1.topic11" (figure 1.6).
- **Fanout Exchange :** c'est une diffusion permettant de distribuer un message à toutes les files d'attente disponibles (figure 1.7).
- **Header Exchange :** le message est acheminé vers les files d'attente en fonction de ses valeur d'en-tête. Le producteur ajoute des valeurs sous forme de paire clé-valeur et un opérateur (all ou any), par exemple 'x-match' => 'any',value1 => '40', value2 => '70', dans l'en-tête du message et l'envoie au service d'échange qui test la correspondance tout ou partie des valeurs avec ceux des files d'attente. Une fois la correspondance est trouvée, il achemine le message vers la file d'attente appropriée (figure 1.8).

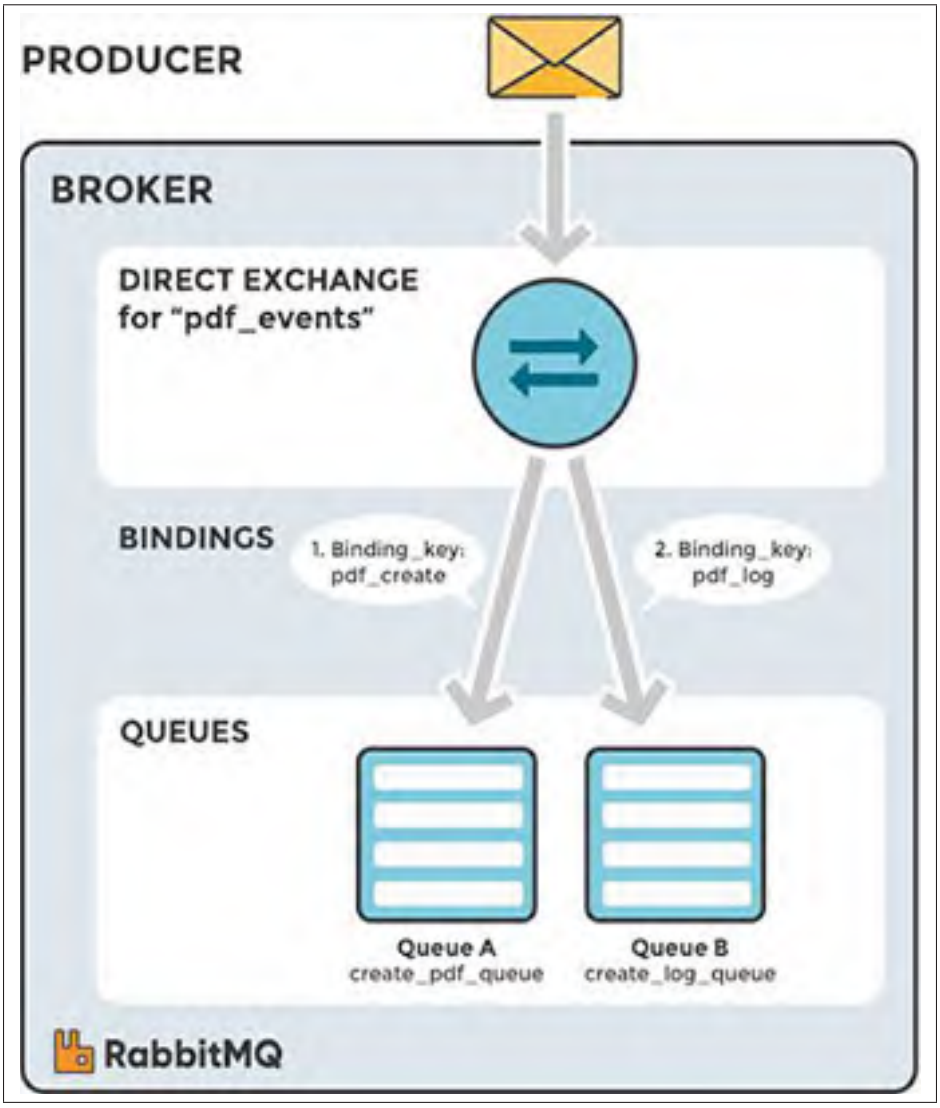


Figure 1.5 RabbitMQ de type direct exchange, référence : RabbitMQ (2019)

1.6 Les techniques d'apprentissage automatique

L'apprentissage automatique ou Machine Learning est basé sur les algorithmes de régression et de classification.

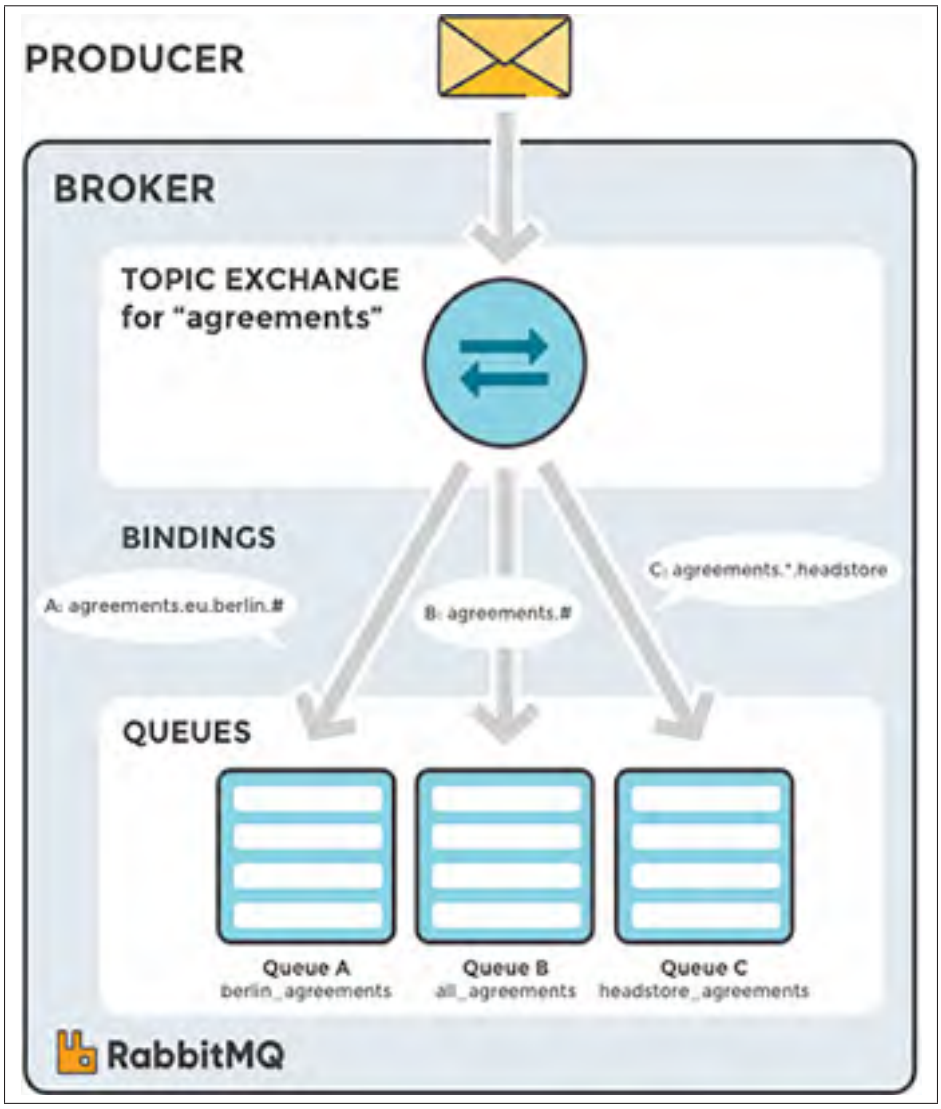


Figure 1.6 RabbitMQ de type topic exchange, référence : RabbitMQ (2019)

1.6.1 Régression

La régression est la prédiction d'une variable quantitative en utilisant une ou plusieurs autres variables à l'aide des algorithmes de régression.

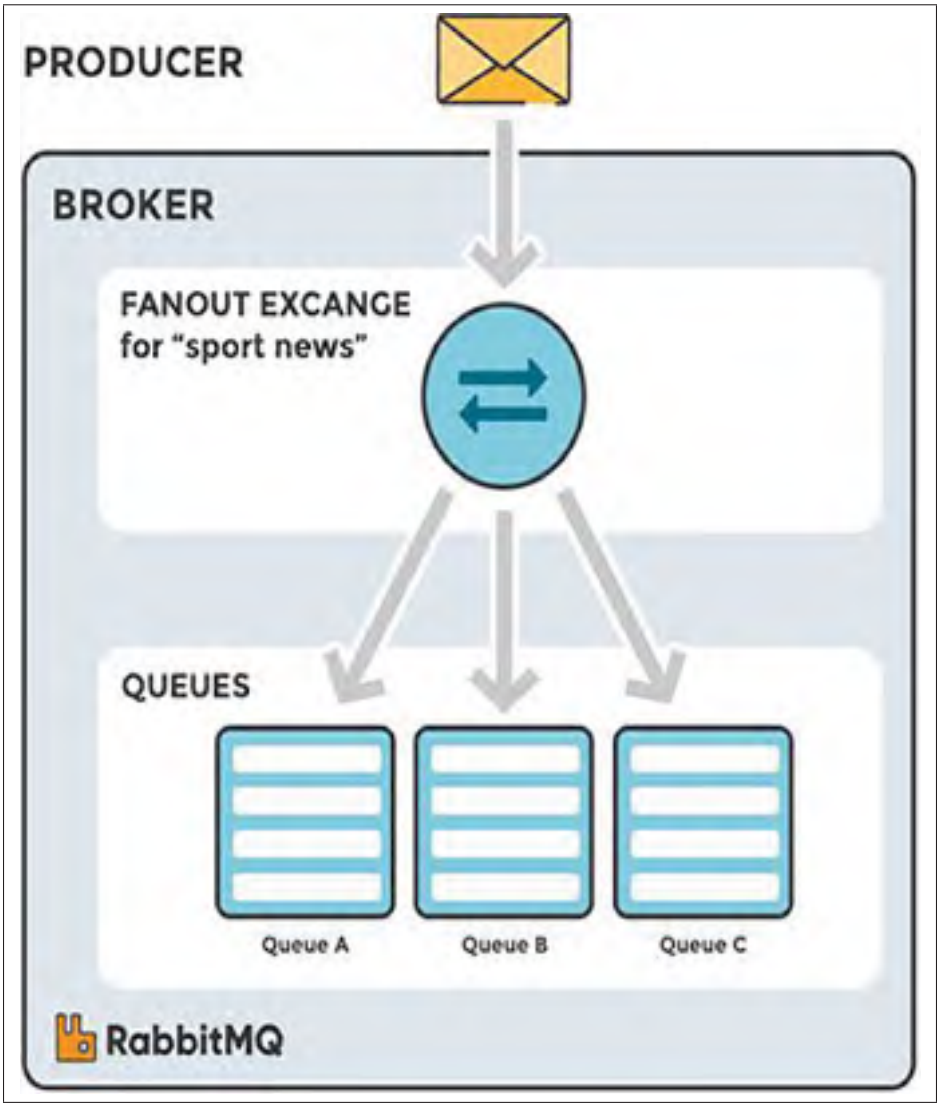


Figure 1.7 RabbitMQ de type fanout exchange, référence : RabbitMQ (2019)

1.6.1.1 Algorithmes de régression

Nous utiliserons les algorithmes de régression dans l'implémentation de filtrage Top-k basé sur le score 3.3. Parmi ces algorithmes nous citons :

- **la régression linéaire :** un modèle qui cherche à établir une relation linéaire entre une variable à prédire et d'autres variables.

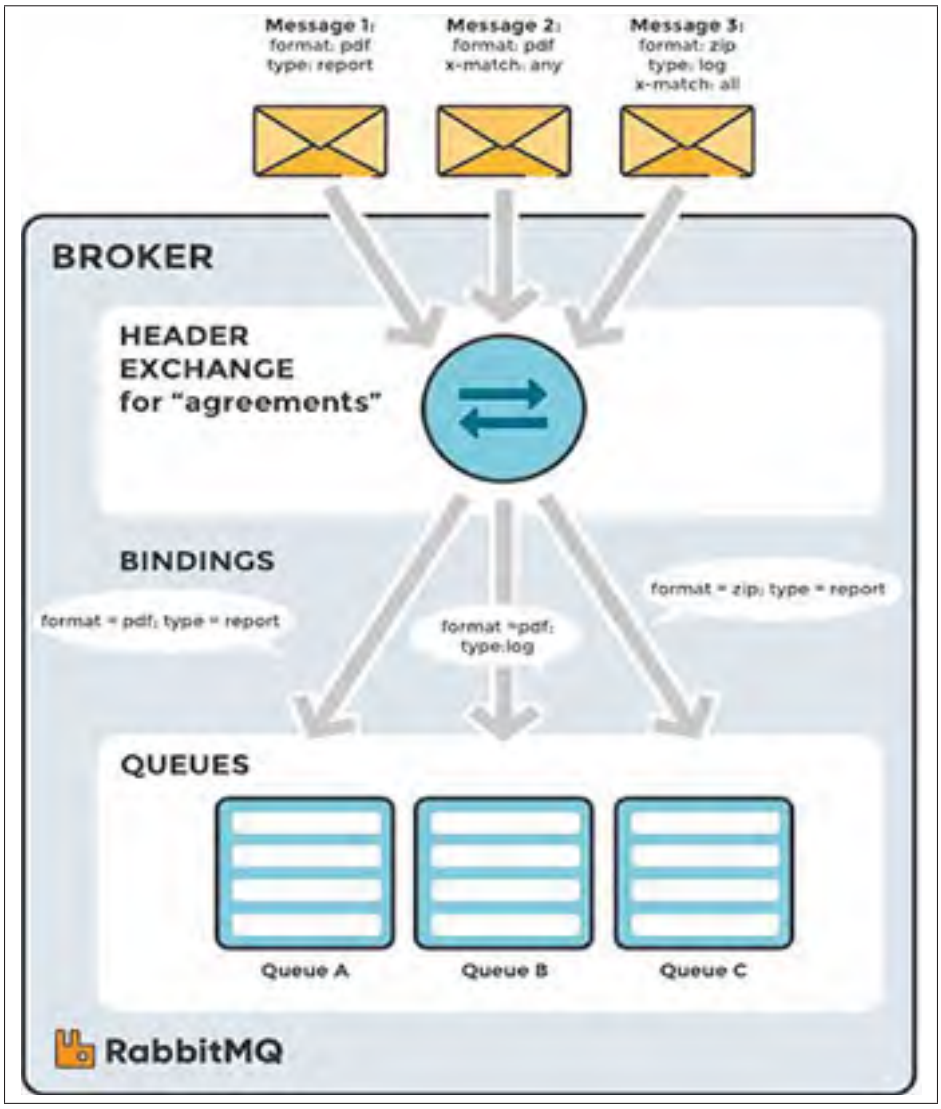


Figure 1.8 RabbitMQ de type header exchange, référence : RabbitMQ (2019)

- **l'arbre de régression :** il est basé sur l'arbre de décision pour prédire une variable cible.
- **K voisins les plus proches (KNN) - régression :** il consiste à prendre en compte un nombre *k* (choisi par l'utilisateur) de voisins selon une distance déterminée pour estimer une variable.

1.6.1.2 Métriques de régression

Afin de choisir l'algorithme adéquat dans un tel cas, nous comparons les algorithmes de régression en utilisant des métriques de régression. Parmi ces métriques on a :

- **Mean Absolute Error (MAE) :** la moyenne des différences entre les valeurs estimées et les valeurs réelles. 0 représente un ajustement parfait.
- R^2 **:** une indication de la qualité de l'ajustement d'un ensemble de valeurs estimées par rapport aux valeurs réelles. C'est la proportion de variance de la variable dépendante qui est prévisible à partir de la ou des variables indépendantes. Il varie entre 0 (non ajusté) et 1 (ajustement parfait).

1.6.2 Classification

La classification est la prédiction d'une variable qualitative, souvent oui/non en utilisant un ou plusieurs autres variables à l'aide des algorithmes de classification tel que feedforward que nous allons l'utiliser dans la section 3.3.2.2.

1.7 Conclusion

Dans ce chapitre, nous avons présenté certaines notions de base relatives à notre travail qui sont : Real-Time Bidding, la spécification OpenRTB, l'enchère de Vickrey, la base de données iPinYou, Publish/Subscribe tels que kafka et RabbitMQ et les techniques d'apprentissage automatique, ce qui permet de mieux comprendre le reste de ce mémoire.

CHAPITRE 2

REVUE DE LITTÉRATURE

Dans cette section, nous présentons les différents travaux liés à notre projet. Nous commençons par Real-Time Bidding et Header Bidding dans le cadre de la publicité en ligne. Ensuite, nous citons de différents travaux portant sur Publish/Subscribe.

2.1 Publicité en ligne

La publicité en ligne était basée sur l'achat direct jusqu'à l'apparition de Real-Time Bidding et Header Bidding. Dans l'achat direct, l'éditeur et l'annonceur s'accordaient dès le début sur le prix payé, le public cible et le nombre d'espaces publicitaires à livrer dans une certaine période de temps, dans le cadre d'un contrat de réservation (Sayedi (2018)). Cette approche statique peut réduire le gain des éditeurs et n'optimise pas les prix payés.

2.1.1 Real-Time Bidding

Selon Yuan *et al.* (2014), la plupart des travaux de recherche dans le domaine de RTB sont orientées vers l'optimisation des algorithmes de DSP pour calculer le prix de l'offre, par exemple dans Wang, Zhang & Yuan (2016), les auteurs proposent de prédire la réponse de l'utilisateur pour estimer la probabilité de ses clics et ses conversions à l'aide de modèles linéaires (régression logistique et régression bayésienne probit) et modèles non linéaires (machines de factorisation, modèle d'arbre dégradé, deep learning), puis l'utiliser dans le calcul d'offre. En plus, dans Lee, Jalali & Dasdan (2013), les auteurs suggèrent un algorithme qui essaie de sélectionner des espaces publicitaires de haute qualité ayant un taux estimé de clics (CTR), d'action ou de conversion (AR) élevé. Dans Zhang & Zhang (2014), les auteurs intègrent le concept d'efficacité de l'utilisation du budget dans la stratégie d'enchères de DSP, pour gagner autant d'espaces publicitaires. Une autre méthode proposée pour les stratégies d'enchères dans Wu, Yeh & Chen (2015), qui consiste à prédire le prix gagnant pour que le DSP propose un prix correct dans l'enchère RTB.

Ces approches sont complémentaires à notre travail, car nous cherchons à réduire les frais de communication liés à l'échange des réponses lors des enchères en optimisant la façon dont l'AdX sélectionne les DSPs à contacter. Une fois sélectionnés, les DSPs peuvent utiliser les stratégies d'enchères proposées dans les travaux de littérature existants pour décider de leurs propres offres. Ces travaux peuvent potentiellement réduire les temps des réponses des DSPs, et par la suite permettre à l'AdX de recevoir les réponses avant le délai d'expiration (t_{max}). De plus, nous mettons en œuvre certaines de ces stratégies d'enchères dans notre évaluation afin de générer une charge réaliste pour les réponses aux enchères (chapitre 4.1).

2.1.2 Header Bidding

Le Header Biding ou la pré-enchère est une technique de publicité en ligne qui permet aux éditeurs d'offrir une espace publicitaire à de nombreux AdXs avant de lancer l'enchère de RTB. Chaque AdX propose un prix afin d'obtenir l'espace d'annonce pour ses DSPs. Ensuite, l'éditeur choisit l'offre la plus élevée et envoie les informations de l'espace publicitaire au AdX gagnant qui commence l'enchère RTB (Qin, Yuan & Wang (2017)). Ce processus aide les éditeurs à accroître leurs bénéfices en choisissant le prix optimal et permet aux annonceurs d'enchérir sur plusieurs espaces d'annonces au lieu de les vendre par contrats de réservation selon Sayedi (2018).

Le Header Bidding est complémentaire à notre solution puisqu'un AdX doit mener une enchère pour ses propres DSPs afin de proposer un prix à l'éditeur et de concourir avec les autres AdXs. Ainsi, il peut utiliser notre solution pour sélectionner dynamiquement l'ensemble des DSPs à contacter pour une telle enchère RTB.

2.2 Publish/Subscribe augmenté

Plusieurs travaux dans la littérature ont parlé des systèmes Pub/Sub, et ont proposé de nouveaux modèles et fonctionnalités permettant de les optimiser et évoluer.

On peut classer ces travaux selon les catégories suivantes :

- Modèles de correspondance
- Modèles de routage
- Fonctionnalités expressives

2.2.1 Modèles de correspondance

Les systèmes Pub/Sub diffèrent dans la manière d'expression des abonnements selon des modèles de correspondances spécifiques tels que topic-based, content-based, type-based et graph-based selon Eugster *et al.* (2003).

- **Topic-based ou basé sur le sujet :** (figure 2.1) où les publications sont classés selon des sujets identifiés par des mots clés, et les abonnés peuvent s'abonner à un sujet spécifique pour recevoir ses publications.

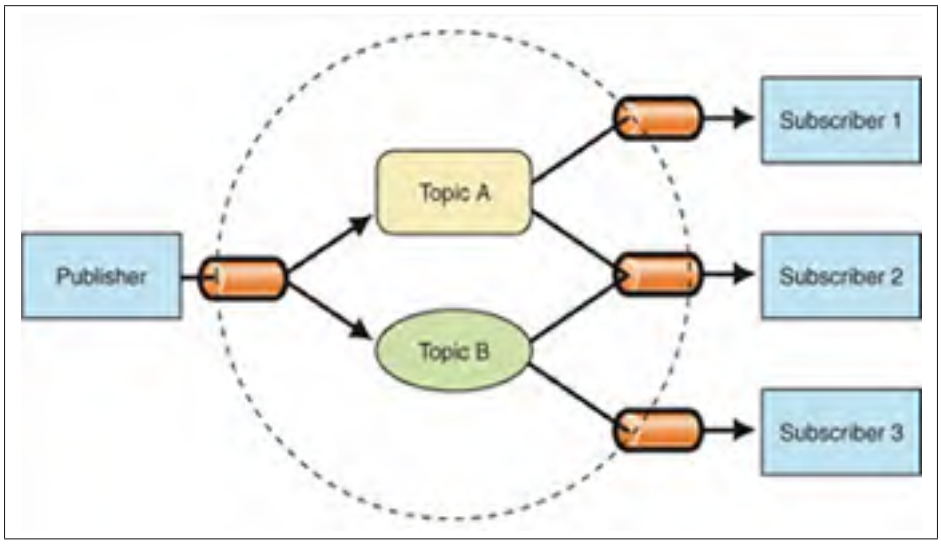


Figure 2.1 Système Publish/Subscribe basé sur le sujet

- **Content-based ou basé sur le contenu :** filtre le contenu des publications à l'aide des prédicats et les abonnés peuvent exprimer leur intérêt à un contenu spécifique. Le système Pub/Sub basé sur le contenu introduit des schémas d'abonnement basés sur un ensemble de paires de propriétés (nom-valeur) et un opérateur de comparaison de base (=, <, >)

selon Canas, Zhang, Kemme, Kienzle & Jacobsen (2017). Par exemple, nous avons deux abonnements Sub1 et Sub2 avec deux paires de propriétés (x op1 val1) et (y op2 val2), qui sont :

$$\text{Sub1} : (x = 3), (y > 4)$$

$$\text{Sub2} : (x < 0), (y > 3)$$

Une fois l'éditeur publie le message Pub :

$$\text{Pub} : (x = 3), (y = 5)$$

Pub/Sub le remet à son abonné intéressé Sub1 comme le montre la figure 2.2.

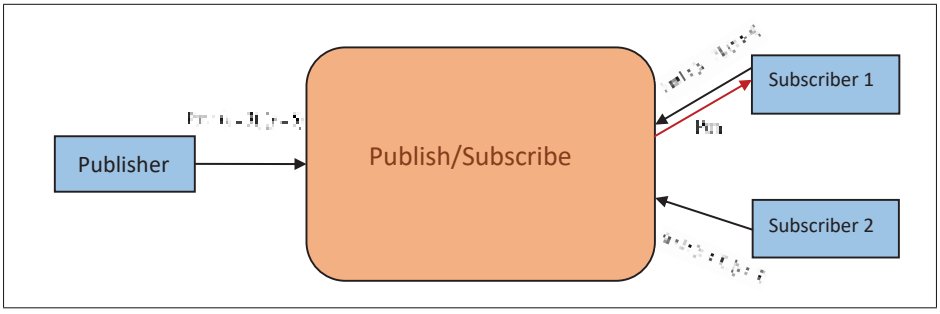


Figure 2.2 Système Publish/Subscribe basé sur le contenu

- **Type-based ou basé sur le type :** (figure 2.3) filtre les évènements en fonction de leur types.
- **Graph-based ou basé sur le graphe :** (figure 2.4) un autre type du modèle de correspondance introduit par Cañas, Pacheco, Kemme, Kienzle & Jacobsen (2015) qui utilise le graphe. Chaque nœud du graphe est un courtier qui prend en charge un abonnée ou/et un éditeur et responsable de lui servir en tenant compte seulement de ses courtiers voisins.

Dans notre travail, nous choisissons le type basé sur le contenu, car il est plus expressif au contraire de celui basé sur le sujet qui apparaît plus statique et primitif.

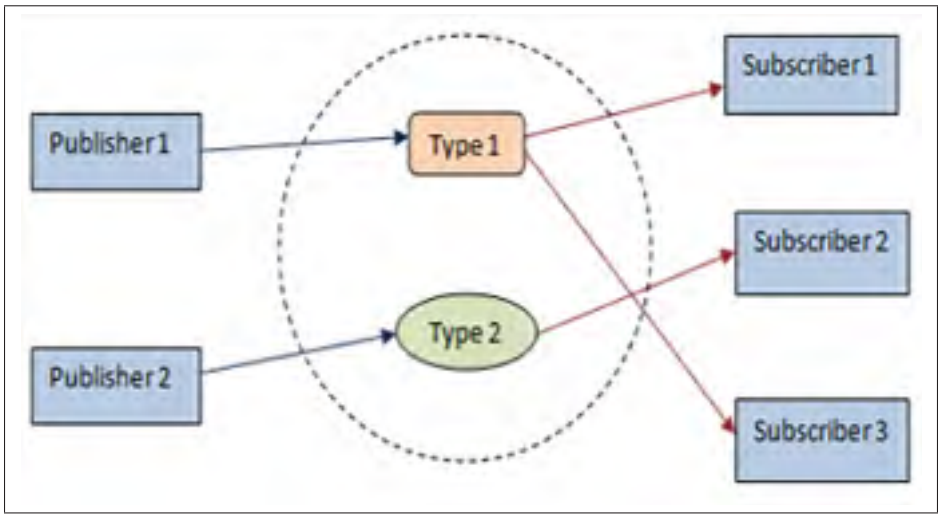


Figure 2.3 Système Publish/Subscribe basé sur le type

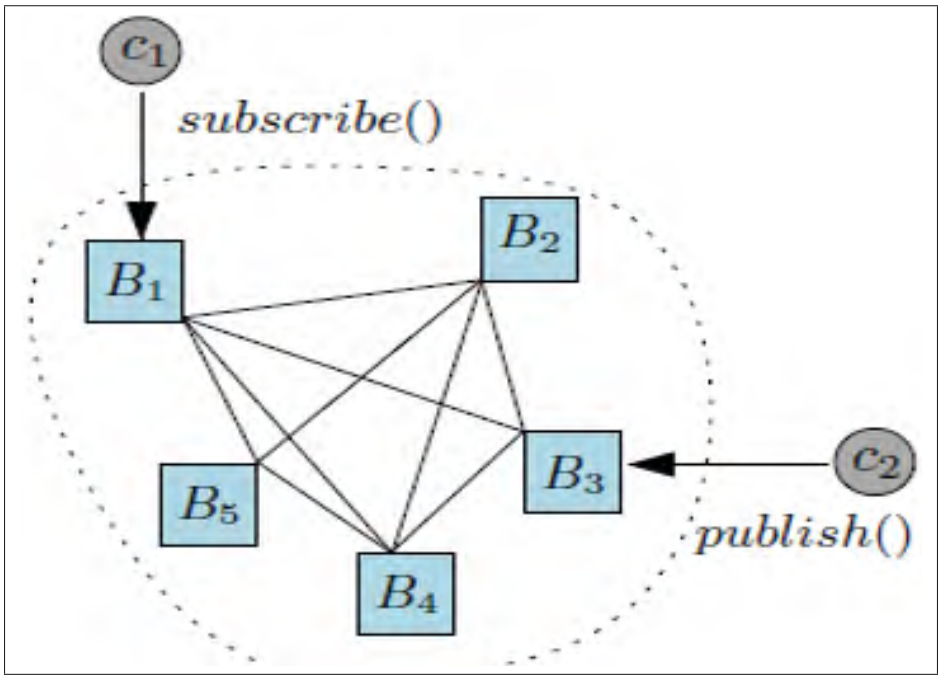


Figure 2.4 Système Publish/Subscribe basé sur le graphe

2.2.2 Modèles de routage

Les systèmes Pub/Sub diffèrent aussi dans la manière de routage des publications vers leurs abonnés correspondants. Dans la littérature, nous trouvons plusieurs exemples de systèmes Pub/Sub ayant de différents modèles de routage

- **Destination-based :** les évènements sont acheminés en fonction de leurs adresses de destinations finales. Par exemple : MEDYM (Cao & Singh (2005)) et MERC (Ji, Ye, Wei & Jacobsen (2015))
- **Rendezvous-based :** chaque sujet a un point de rendez-vous auquel tous les abonnements et les évènements correspondants sont acheminés. Par exemple : SCRIBE (Rowstron, Kermarrec, Castro & Druschel (2001)), HERMES (Pietzuch & Bacon (2002))
- **Advertisement-based :** (figure 2.5) chaque éditeurs annonce sa publication avant de l'envoyer en formant un chemin. Ensuite, les abonnements suivent le chemin inverse pour s'abonner à la publication correspondante à cette annonce. Enfin, la publication suit ce chemin pour aller vers l'abonné intéressé. Par exemple : HERMES (Pietzuch & Bacon (2002))

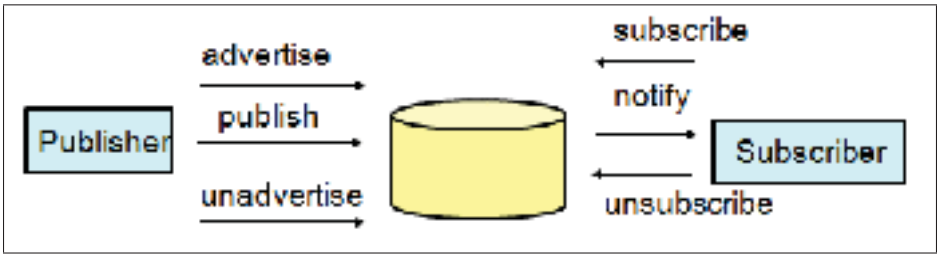


Figure 2.5 Système Publish/Subscribe advertisement-based

- **Subscription-based :** (figure 2.6) les abonnés désirant recevoir une telle publication doivent envoyer un abonnement d'abord, pour la recevoir une fois elle se produit. Exemple : SIENA (Carzaniga, Rosenblum & Wolf (2001))

Dans notre projet, le modèle de routage le plus approprié est le subscription-based, puisqu'il permet aux DSPs d'envoyer les intérêts de leurs annonceurs à l'AdX afin de recevoir les demandes d'offres seulement sur des espaces publicitaires intéressants.

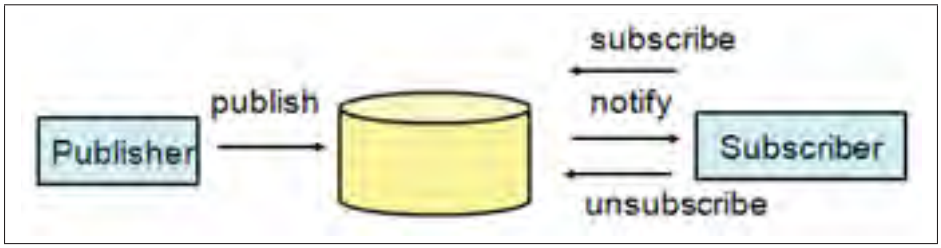


Figure 2.6 Système Publish/Subscribe subscription-based

2.2.3 Fonctionnalités expressives

Dans la littérature, plusieurs auteurs ont ajouté des fonctionnalités au système Pub/Sub afin de l'optimiser, tels que l'agrégation, l'abonnement évolutif, l'abonnement paramétrique et le filtrage Top-k.

2.2.3.1 Agrégation

Selon Jacobsen, Pandey, Vitenberg, Zhang & Weiss (2014), l'agrégation permet de regrouper les publications et les répartir dans des fenêtres d'agrégation afin d'éviter de diffuser un grand nombre de publications.

2.2.3.2 Abonnement évolutif

Selon Canas *et al.* (2017), l'abonnement évolutif permet de mettre à jour de manière autonome un abonnement sans l'intervention de la part d'abonné, ce qui élimine le fait de se désabonner et réabonner à chaque fois.

2.2.3.3 Abonnement paramétrique

Selon Jayaram, Jayalath & Eugster (2010), l'abonnement paramétrique est un abonnement contenant une expression de prédicat qui dépend du temps.

2.2.3.4 Filtrage Top k

Top-k est une fonctionnalité de Pub/Sub permettant d'envoyer une publication uniquement aux k meilleurs abonnés. Le filtrage Top-k calcule le score de chacun, puis les classe par ordre décroissant et choisit les k meilleurs abonnés. Le filtrage Top-k a été utilisé dans certains travaux précédents, comme dans Shraer, Gurevich, Fontoura & Josifovski (2014), où les auteurs proposent d'afficher seulement les Top-k tweets sur chaque story social dans un site Web proposant un nombre élevé de d'actualités et de vues, afin de réduire les coûts de ressources. La sélection des tweets est basée d'abord sur le modèle Pub/Sub puis sur une fonction de score évaluant le contenu et la récence. D'une autre manière, Top-k est défini dans Zhang, Sadoghi, Muthusamy & Jacobsen (a) pour fournir une publication uniquement aux k abonnés les mieux classés en utilisant la notion de couverture des abonnements (covering) pour les applications à grande échelle. Le filtrage Top-k est également utilisé dans Zhang, Sadoghi, Muthusamy & Jacobsen (2013) pour réduire le nombre de notifications envoyées dans les réseaux sociaux. Dans Chen, Cong, Cao & Tan (2015), les auteurs proposent une nouvelle solution pour gérer un grand nombre de requêtes de type Temporal Spatial-Keyword Top-k TaSK dans le flux d'objets de type géographique. Mais, le filtrage Top-k n'a été jamais utilisé auparavant dans le contexte de la publicité RTB selon notre connaissance. Dans notre travail, nous l'utilisons comme méthode pour sélectionner les DSPs les plus offrants pour une publication donnée.

2.3 Conclusion

Dans cette section, nous avons présenté un bref résumé des solutions les plus pertinentes par rapport à notre problème de recherche. D'abord, les solutions proposées dans le domaine de RTB sont presque toutes dirigées vers l'optimisation des algorithmes ou des stratégies d'enchères des DSPs afin de garantir les profits de ses annonceurs, sans prendre en considération l'AdX qui est responsable du bon déroulement d'enchère et le choix du meilleur prix dans un temps précis. Dans ce mémoire, nous intégrons dans AdX un système Pub/Sub basé sur le contenu (modèle de correspondance) et subscription-based (modèle de routage), afin de filtrer les requêtes d'enchères par rapport aux abonnements des DSPs, pour réduire ses coûts de calcul et de communication,

ainsi que pour diminuer le délai d'attente. Notre travail est parfaitement cohérent avec les stratégies d'enchères proposées dans la littérature ainsi que la publicité Header Bidding. Ensuite, nous proposons d'ajouter une extension de filtrage Top-k dans AdX, qui n'a été jamais utilisé dans le contexte de publicité RTB selon notre connaissance, pour lui permettre de sélectionner un sous-ensemble de DSPs les plus compétitifs.

CHAPITRE 3

SOLUTION PROPOSÉE

Dans ce chapitre, nous présentons d'abord notre nouvelle architecture du système RTB avec Pub/Sub. Ensuite, nous définissons notre problème du nombre optimal de réponses aux enchères. Enfin, nous proposons d'ajouter une extension de filtrage Top-k au Pub/Sub comme solution heuristique qui résout ce problème.

3.1 Intégration de Publish/Subscribe basé sur le contenu dans Real-Time Bidding

Comme nous avons vu dans la section 1.3, le DSP peut retourner des réponses vides ou sans offres à AdX, ce qui augmente les coûts de communication liés à l'échange des requêtes d'enchères non pertinentes et des réponses vides. Selon notre analyse de la base de données iPinYou, nous avons remarqué que la plupart des réponses vides sont causées par les raisons : appareil incompatible ou utilisateur non ciblé (codes 6 et 8 dans le tableau 1.3).

Nous intégrons le modèle Publish/Subscribe basé sur le contenu dans AdX afin d'améliorer le système RTB. Il permet de filtrer les requêtes d'enchères et les envoyer uniquement à leurs DSPs intéressés et par la suite éviter les réponses vides.

La figure 3.1 montre l'architecture du nouveau processus de RTB. Par rapport au processus original vu dans la section 1.1, nous avons modifié les étapes suivantes :

(1a) Avant de recevoir la requête d'enchère, les DSPs envoient à AdX (équipé d'un courtier Pub/Sub) les caractéristiques intéressantes de l'espace publicitaire et le public cible sous forme d'un abonnement basé sur le contenu comme suit :

$$\textit{Abonnement : (device, =, val1), (user\_profile, =, val2)}$$

device : appareil de l'utilisateur à travers lequel il interagit, il peut faire référence à un mobile, un ordinateur de bureau, un décodeur ... Cette variable est appelée *device* dans la spécification

OpenRTB (IAB (2016)) et *User-Agent* dans la base de données iPinYou (Liao *et al.* (2014)). *user _profile :* ce sont les mots-clés, intérêts, genre, âge, etc. de l'utilisateur. Cette variable est appelée *user* dans la spécification OpenRTB (IAB (2016)) et *UserProfileIDs* dans la base de données iPinYou (Liao *et al.* (2014)).

(2) Une fois qu'un internaute visite une page Web, l'éditeur remet à AdX une requête d'annonce contenant notre publication comme suit :

Publication : (device, val1), (user _profile, val2)

(3) Pour chaque requête d'annonce entrante, AdX génère une requête d'enchère ou requête d'enchère comme suit :

Requête d'enchère : (enchères, site, appareil, données utilisateur)

AdX la filtre selon les abonnements stockés et (4) l'envoie à leurs DSPs intéressés, qui calculent les prix des offres en fonction de leurs campagnes. Alors que les DSPs ayant des abonnements non correspondants ne reçoivent pas la requête d'enchère.

Le système Pub/Sub permet à chaque DSP d'envoyer plusieurs abonnements si nécessaire. De plus, les abonnements peuvent être modifiés à tout moment pour refléter les intérêts mis à jour du ses annonceurs.

3.2 Problématique

Maintenant, nous nous concentrons sur le problème de filtrage des DSPs au-delà de ceux qui ne sont pas intéressés. La diminution des DSPs à contacter est judicieuse pour optimiser notre système spécifiquement en cas de surcharge tout en conservant un prix élevé pour chaque enchère, en attendant juste assez longtemps pour collecter ce prix gagnant (qui est le deuxième meilleur

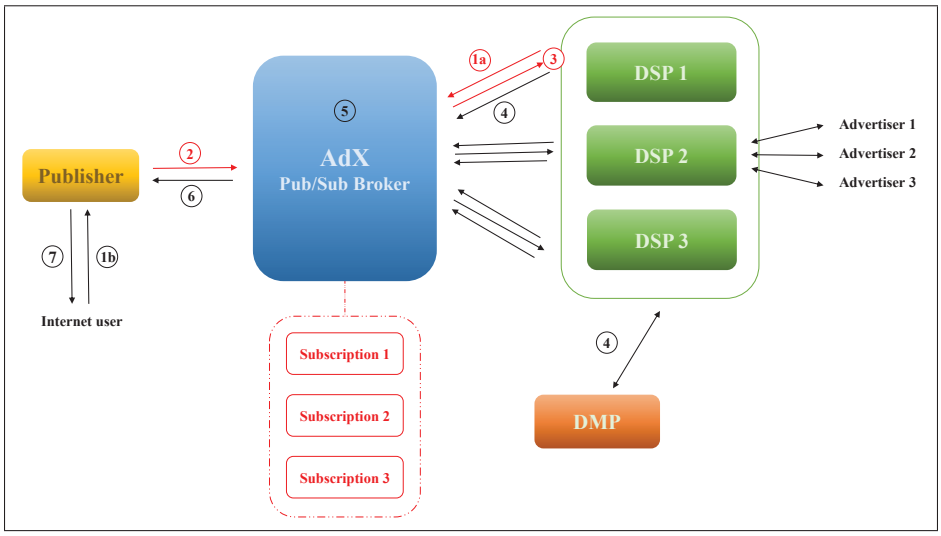


Figure 3.1 Le nouveau processus de RTB avec Pub/Sub basé sur le contenu

prix selon l'enchère de Vickrey). La diminution du temps d'enchère total est très importante puisqu'elle optimise l'utilisation des ressources du système, car plus qu'une enchère prend de temps, plus les ressources (par exemple la RAM) sont bloquées. Cela signifie que les DSPs avec des prix d'offres non compétitifs et des temps des réponses lents doivent également être filtrés par notre solution. Donc, nous définissons un modèle général qui peut capturer ces deux paramètres à l'aide d'un modèle de coût. Nous formalisons notre problème d'optimisation qui permet de sélectionner un nombre minimal de réponses aux enchères en conservant un deuxième meilleur prix élevé et un délai d'attente minimal, comme suit :

Étant donné un ensemble de réponses aux enchères $BR = \{(b_1, t_1), (b_2, t_2), ..., (b_n, t_n)\}$, *où* n *est le nombre de DSPs,* b_i *est le prix offert,* t_i *est le temps de réponse,* BR^c *est un sous-ensemble de* BR *et* BR^d *est un sous-ensemble de* BR^c

$$\textbf{Minimize } |BR^c|, \textit{ where } BR^c \subseteq BR$$

$$\textbf{Subject to } U(BR^c) \geq U(BR^d), \forall BR^d \subseteq BR$$

$$U(BR_i) = w_1 \times second\_price(BR_i) + w_2 \times (t_{max} - max\_time(BR_i))$$

$$second\_price(BR_i) = max_{j \neq i} b_j, \textit{ where } b_i > max_{j \neq i} b_j, \forall (b_i, t_i), (b_j, t_j) \in BR_i$$

$$max\_time(BR_i) = max(t_i), \forall (b_i, t_i) \in BR_i$$
$$w_1 + w_2 = 1$$

Dans la formule ci-dessus, $U(BR)$ représente l'utilité d'un sous-ensemble de réponses à l'enchère, qui est calculée en tenant compte des deux paramètres objectifs : le prix payé (deuxième meilleur prix d'offre) et le temps de l'enchère (le temps de réponse la plus lente reçue). Ces deux mesures sont normalisées à l'aide des paramètres de poids w_1 et w_2, qui peuvent être ajustés en fonction de l'application. L'ensemble optimal des réponses aux enchères choisies est celui qui fournit la meilleure utilité avec un minimum de DSPs contactés.

Étant donné un oracle avec une connaissance parfaite, qui peut retourner avec précision un nombre minimal des réponses aux enchères ayant des meilleurs prix et temps de réponses minimales, nous pouvons prouver que la solution optimale a toujours la taille deux :

Theorem 2. *Le nombre de DSPs choisi est toujours égal à 2, ($|BR^c| = 2$).*

Preuve par contradiction On suppose que l'ensemble optimal de réponses à l'enchère choisies est $BR^c = (b_i, t_i), (b_s, t_s), (b_w, t_w)$. D'où $|BR^c| = 3$, sans perte de généralité et $U((b_i, t_i), (b_s, t_s), (b_w, t_w))$ est maximum.

cas 1 : Si $b_i < second\_price(BR^c)$, alors $b_i < b_s < b_w$ et $t_i \leq max\_time(BR^c) \Rightarrow (b_i, t_i)$ peut être supprimé sans affecter l'utilité, car il ne contribue ni à augmenter le prix gagnant (deuxième meilleur prix), ni à augmenter le temps de l'enchère.

cas 2 : $\forall b, b \geq second\_price(BR^c)$, alors $\exists b_s$ et $b_w \geq second\_price \Rightarrow (b_i, t_i)$ peut également être supprimé, car cela signifie qu'au moins 2 des 3 offres ont la même valeur, donc la suppression de l'une des trois n'affectera pas la valeur du deuxième meilleur prix. La suppression d'une enchère également ne peut pas augmenter le temps de l'enchère restant.

Donc, dans les deux cas : $U((b_i, t_i), (b_s, t_s), (b_w, t_w)) = U((b_s, t_s), (b_w, t_w))$, alors $|BR^c| = |BR^c - (b_i, t_i)| = 2 < 3$.

Par conséquent, notre problème peut être défini comme étant une sélection de 2 réponses à l'enchère à partir n réponses, où l'utilité du sous-ensemble choisi est maximale parmi tous les autres sous-ensembles de taille 2. Il s'agit donc d'un problème de combinaison avec $\binom{n}{2}$. Cette combinaison peut être résolue en temps quadratique $O(n^2)$ selon Oppen (1980).

3.3 Filtrage Top-k

Dans la section précédente, nous avons démontré que AdX doit sélectionner de façon optimale 2 DSPs pour chaque demande d'enchère. En pratique, cela n'est pas possible sans une parfaite connaissance des futurs prix des offres et des temps dse réponses. Par conséquent, nous proposons l'utilisation du filtrage Top-k comme solution générale pour sélectionner un sous-ensemble de DSPs avec une taille fixe k, selon une fonction de score qui modélise la fonction d'utilité de la section précédente. Dans des circonstances idéales, notre solution Top-k donnerait des résultats optimaux si $k = 2$ et la fonction de score correspond parfaitement à la fonction d'utilité $U(BR)$.

La figure 3.2 illustre notre solution proposée avec le Pub/Sub et Top-k. Pour chaque requête d'enchère arrivée b, AdX teste sa correspondance avec les n DSPs à l'aide du service Pub/Sub pour obtenir les $m \leq n$ DSPs correspondants à b, puis sélectionne $k \leq m$ DSPs à l'aide du filtrage Top-k et leur distribue la requête d'enchère b.

Figure 3.2 Flux de traitement avec filtrage Pub/Sub et Top-k

La figure 3.3 montre en détail comment fonctionne le filtrage Top-k basé sur le score. Les performances de chaque DSP sont prévues afin de calculer leurs scores. Ils sont ensuite triés afin de sélectionner les k meilleurs DSPs.

Le score d'un DSP_i pour une requête donnée b est calculé selon la formule suivante :.

Figure 3.3 Diagramme de Top-k basé sur le score

$$score(DSP_i, b) = w_1 \times predicted\_performance(DSP_i, b) - w_2 \times won\_bids(DSP_i) \quad (3.1)$$

et $predicted\_performance(DSP_i, b)$ est calculé comme suit :

$$predicted\_performance(DSP_i, b) = \frac{predicted\_price(DSP_i, b)}{predicted\_time(DSP_i, b)} \quad (3.2)$$

La formule 3.2 est similaire à la formulation théorique, car elle considère le prix d'offre et de temps de réponse à la fois. Mais en l'utilisant uniquement, il y a des DSPs qui ne seront jamais ou moins contactés. Pour introduire une certaine équité dans le système et par la suite permet à l'AdX de gagner la confiance de ses clients, nous avons ajouté à la formule 3.1 le paramètre *#wonBids* qui indique le nombre d'offres déjà gagnées par un DSP_i. Les métriques de score sont normalisées à l'aide des paramètres de poids w_1 et w_2, qui peuvent être ajustés en fonction de l'application.

3.3.1 Modèle de prédiction des scores

Afin de prédire la performance d'un DSP, nous avons besoin d'estimer son prix d'offre et son temps de réponse en utilisant des données historiques des requêtes des enchères (par exemple : la base de données iPinYou) qui fournissent des informations sur l'espace publicitaire et l'utilisateur (variables indépendantes), ainsi que les réponses correspondantes des DSPs, incluant les prix des offres et les temps des réponses (variables dépendantes). Nous utilisons l'analyse de régression

pour prédire le prix d'offre et le temps de réponse comme étant deux variables continues dans $\mathbb{R}_{\geq 0}$.

Tout d'abord, nous sélectionnons les attributs de la base de données iPinYou à utiliser pour la régression, en utilisant la méthode du filtre (KAUSHIK). Cette méthode consiste à étudier les corrélations entre chaque pair d'attributs à l'aide de la matrice de corrélation (tableau 3.1), puis supprimer un attribut dans chaque pair fortement corrélé (coefficient de corrélation |c| > 0,5 coloré en rouge), on choisit celui le moins corrélé avec le prix d'enchère et le temps de réponse.

Tableau 3.1 Matrice de corrélation de la base de données iPinYou

	(f1)	**(f2)**	**(f3)**	**(f4)**	**(f5)**	**(f6)**	**(f7)**	**(f8)**	**(f9)**	**(f10)**	**(f11)**
Timestamp (f1)	1.00	-0.28	0.03	-0.04	0.18	0.18	-0.08	-0.34	-0.30	0.09	-0.03
AdExchange (f2)	-0.29	1.00	-0.13	-0.12	0.08	0.07	0.03	0.14	0.15	0.10	0.04
AdSlotHeight (f3)	0.29	-0.13	1.00	0.60	-0.19	-0.19	0.16	-0.01	0.03	-0.30	-0.01
AdSlotWidth (f4)	-0.04	-0.12	0.60	1.00	-0.18	-0.17	0.16	0.005	-0.01	-0.16	0.01
RegionID (f5)	0.18	0.08	-0.19	0.18	1.00	0.99	0.005	-0.16	-0.14	0.19	0.04
CityID (f6)	0.18	0.07	-0.19	-0.17	0.99	1.00	0.01	-0.17	-0.15	0.18	0.04
AdSlotVisibility (f7)	-0.08	0.03	0.16	0.16	0.06	0.01	1.00	-0.09	-0.09	-0.11	0.02
AdSlotFloorPrice (f8)	-0.34	0.14	-0.01	0.005	-0.16	-0.17	-0.09	1.00	0.97	-0.02	-0.01
BiddingPrice (f9)	-0.30	0.15	-0.03	-0.01	-0.14	-0.15	0.09	0.97	1.00	0.01	-0.01
ResponseTime (f10)	0.09	0.11	-0.29	-0.16	0.19	0.18	-0.11	-0.02	0.01	1.00	0.01
UserProfileIDs (f11)	-0.30	0.04	-0.01	0.01	0.04	0.04	0.02	-0.01	-0.01	0.01	1.00

Nous obtenons donc les attributs suivants : Timestamp, AdExchange, AdSlotHeight, CityID, AdSlotVisibility, AdSlotFloorPrice et UserProfileIDs.

Pour prédire le prix d'enchère et le temps de réponse, nous implémentons avec Python les algorithmes de régression vue dans la section 1.6.1.1 : régression linéaire, arbre de régression et K voisins les plus proches (pour régression) sur notre base de données générée dans la section 4.3 et l'historique des réponses des DSPs, et nous comparons leurs résultats en utilisant Python pour avoir les métriques de régression vu dans la section 1.6.1.2 afin de choisir le meilleur algorithme permettant de donner les résultats les plus précises. Nous obtenons les résultats de Python cités dans le tableau 3.2. Nous remarquons que la régression linéaire donne de meilleurs résultats

pour le prix d'offre avec $MAE \simeq -3.50$ et $R^2 \simeq 0.88$, et KNN pour temps de réponse avec $MAE \simeq -6.54$ et $R^2 \simeq -13.57$. Par conséquent, nous mettons en œuvre ces deux techniques dans notre évaluation

Tableau 3.2 Les résultats des métriques de régression

Technique de régression	**Métrique**	**MAE**	**R^2**
Régression linéaire	prix d'enchère	-3.50	0.88
	temps de réponse	-8.60	-27.48
Arbre de régression	prix d'enchère	-4.37	0.85
	temps de réponse	-6.84	-38.87
KNN (régression)	prix d'enchère	-35.61	-0.29
	temps de réponse	-6.54	-13.57

3.3.2 Modèles de prédiction discrets

Le filtrage Top-k basé sur le score nécessite la prédiction de deux variables continues : le prix d'offre et le temps de réponse, mais comme nous le verrons dans l'évaluation (section 6), la prédiction des variables continues est un peu lent ce qui a un impact négatif sur la latence de bout en bout de chaque enchère. Ainsi que l'objectif principal du filtrage Top-k est uniquement de déterminer les DSPs à contacter afin de lancer une enchère rentable dans un délai minimal, donc on n'a pas besoin de prédire avec précision les DSPs estimant les meilleurs performants, alors que et le mécanisme de score actuel calcule avec beaucoup de précision ce qui dégrade les performances du système. Afin d'accélérer la prédiction, nous proposons donc deux modèles de prédiction utilisant des variables discrètes : basé sur le rang et basé sur la sélection binaire.

3.3.2.1 Top-k basé sur le rang

Cette méthode est basée sur la prédiction du rang (le 1er, 2e, 3e,...) qui est une variable discret, de chaque DSP intéressé par la requête d'enchère en cours (figure 3.4). Tout d'abord, nous utilisons les mêmes données historiques et les scores obtenus pour classer les DSPs selon des rangs. Puis, nous exécutons l'algorithme de régression linéaire pour prédire le rang de chaque

DSP en tenant compte des caractéristiques de la requête d'enchère. Enfin, l'AdX sélectionne k DSPs ayant les meilleurs rangs.

Figure 3.4 Diagramme de Top-k basé sur le rang

3.3.2.2 Top-k basé sur la sélection binaire

Malgré que l'approche précédente utilise une variable discrète, elle effectue toujours beaucoup de calcul, car elle essaie de prédire avec précision le rang de chaque DSP. Nous proposons donc une autre approche basée sur un modèle binaire estimant si un DSP intéressé à l'espace publicitaire en cours, figure parmi les Top-k ou non («oui» ou «non») à l'aide d'analyse de classification (figure 3.5). Nous avons testé cinq types de classificateurs : FeedForward, Bayesian, NEAT, PNN et RBF et nous avons constaté que FeedForward est le plus rapide. Pour former ce modèle, nous utilisons les même données historiques, avec les réponses «oui» si le DSP a été parmi les top k ou «non», mais cette méthode ne garantit pas obtenir exactement k DSPs comme nous allons voir dans le chapitre 6.

Figure 3.5 Diagramme de Top-k basé sur la sélection binaire

3.4 Conclusion

Dans ce chapitre, nous avons proposé d'intégrer un système Pub/Sub basé sur le contenu dans RTB permettant de filtrer les requêtes d'enchères selon les abonnements des DSPs afin de limiter les réponses vides. Puis, nous avons présenté notre problématique à l'aide d'un modèle mathématique visant à optimiser le nombre de réponses aux enchères en diminuant les réponses

non compétitives afin de réduire les coûts de communication et augmentant l'utilité du système. Enfin, pour s'approcher de la solution optimale nous avons proposé une solution étendu qui combine Pub/Sub et Top-k.

CHAPITRE 4

IMPLÉMENTATIONS

Dans ce chapitre nous allons présenter l'implémentation de nos approches ainsi qu'une démonstration visualisant les résultats de nos solutions et les comparant avec l'approche de base OpenRTB. Enfin, nous développons un générateur de requêtes d'enchères ajustable issu de la base de données iPinYou.

4.1 Projet de recherche

Notre projet de recherche consiste à intégrer le système Pub/Sub avec filtrage Top-k dans OpenRTB. Dans ce cadre, nous avons implémenté deux nouvelles solutions utilisant Pub/Sub et Top-k, afin de les évaluer et les comparer avec notre approche de base OpenRTB. Nous avons utilisé l'implémentation en J2EE d'OpenRTB 2.0 [1], mais nous avons ajouté notre propre AdX selon les spécifications de l'API OpenRTB (version 2.5) décrit dans IAB (2016). Nous avons utilisé également Header Exchange de RabbitMQ [2] (Ionescu (2015)). Enfin, nous avons ajouté le filtrage Top-k dans l'AdX et nous le combinons avec RabbitMQ.

4.1.1 Stratégies d'enchères

Nous utilisons trois types de DSP ayant les stratégies d'enchères suivantes :

- **CONSTANT** : l'implémentation de référence fournit un type de DSP simple, qui ajoute une valeur constante au prix de réserve pour chaque demande.
- **RANDOM** : nous avons implémenté notre propre stratégie d'enchères qui choisit une valeur aléatoire entre le prix de réserve et une valeur maximale paramétrique.
- **Below_eCPC** : nous avons mis en œuvre la stratégie discutée dans Zhang & Zhang (2014) où le prix de l'offre est calculé en multipliant le maximum de eCPCs qui est l'ensemble des

[1] https://github.com/openrtb/openrtb2x

[2] https://github.com/rabbitmq/rabbitmq-tutorials/tree/master/java

CPCs (coût effectif par clic) pour chaque campagne divisée par le nombre de clics atteints, avec le CTR (taux de clics) prévu pour l'espace publicitaire.

4.2 Démo

Pour démontrer l'efficacité de nos solutions, nous fournissons un site Web, qui visualise les résultats des enchères dans l'approche de base OpenRTB et nos approches Pub/Sub et Top-k et compare entre eux en termes d'efficacité (prix payé/temps d'enchère), prix payés, temps de traitement et nombre de messages échangés y compris les réponses vides.

Comme nous voyons dans la figure 4.1, le site Web permet à l'utilisateur d'exécuter en temps réel les trois approches avec la base de données générée ajustable en choisissant le nombre d'enchères, le nombre des DSPs participants, leur seed et le paramètre *k* de Top-k. L'utilisateur peut aussi changer les types des DSPs (RANDOM, CONSTANT ou BelowCPC). En plus, il peut modifier dans la formule de score de Top-k en variant les poids de ses paramètres.

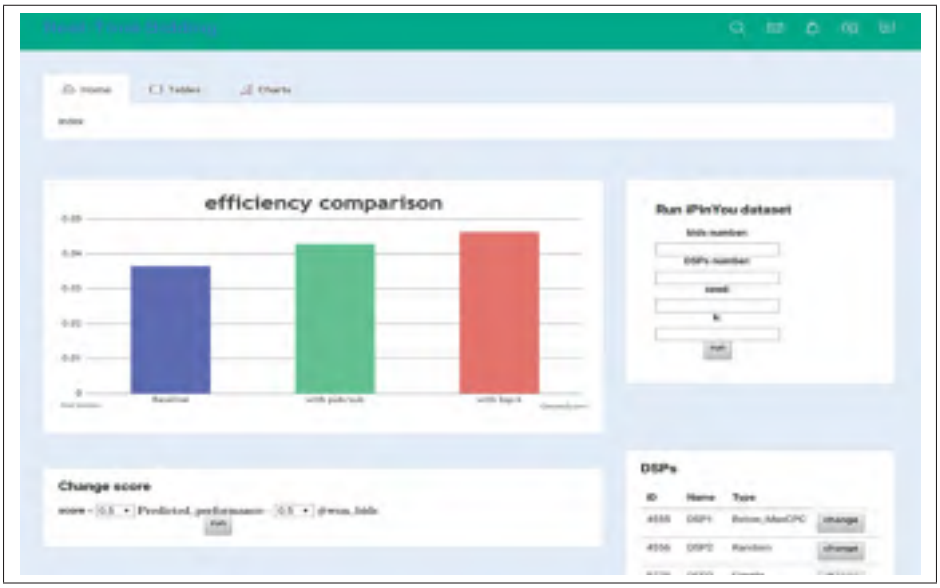


Figure 4.1 Page d'accueil du site web

Après avoir choisi les paramètres d'exécution, l'utilisateur peut voir les résultats des enchères des trois implémentations telle que l'approche de base OpenRTB comme montre figure 4.2. Les

résultats sont affichés sous forme d'un tableau contenant l'id de chaque enchère, son gagnant, le prix payé à la fin et le temps total de traitement, avec la possibilité de visualiser les détails de chaque enchère en sélectionnant l'enchère et cliquant sur "ok" au dessus du tableau.

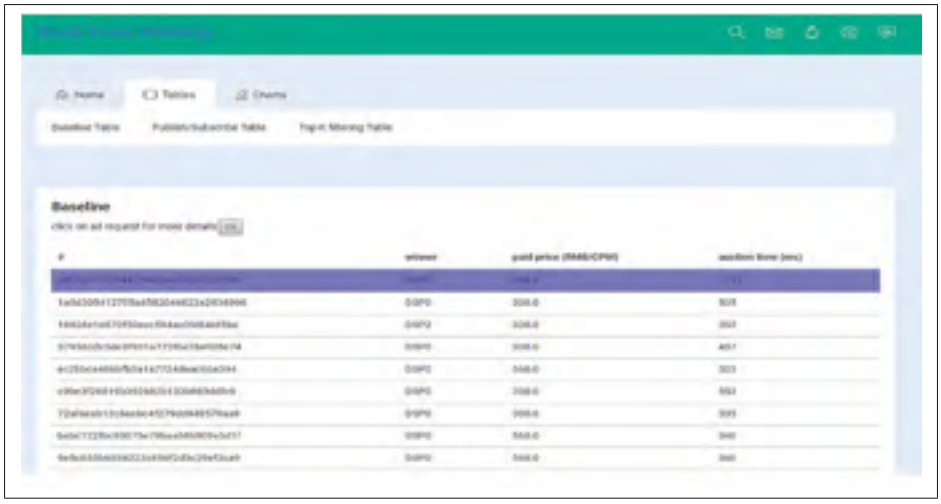


Figure 4.2 Résultats des enchères de l'approche de base OpenRTB

Puis, nous comparons les trois solutions en termes d'efficacité, prix payés, temps de traitement comme nous pouvons le voir dans le figure 4.3 et nombre de messages échangés y compris les réponses vides.

Figure 4.3 Comparaison du temps d'enchère

4.3 Générateur de requêtes d'enchères

Dans ce mémoire, nous décidons d'implémenter un générateur de requêtes d'enchères issu de la base de données d'iPinYou pour l'analyser et le modéliser, ainsi que pour ajuster certains paramètres de la base de données pour l'analyse de sensibilités dans le chapitre 6. Mais avant d'implémenter ce générateur, nous devons étudier la base de données d'iPinYou ainsi que la signification de ses attributs.

4.3.1 Préparation de la base de données iPinYou

Avant d'étudier les distributions des attributs, nous devons préparer la base de données. Cette étape est indispensable, car les données réelles peuvent être incomplètes, bruitées ou incohérentes, ce qui affecte la qualité des données.

On commence tout d'abord par la conversion des données manquantes, qualitatives (chaîne), Null, inconnu (Na) : par exemple pour chaque valeur de visibilité de l'espace publicitaire, nous affectons un nombre (0 : inconnu (Na), 1 : FirstView, 2 : OtherView (SecondView to TenthView)). Ensuite, nous doublons les lignes ayant une liste de UserProfileIDs, chaque ligne doit contenir un seul UserProfileID.

4.3.2 Les distributions de la base de données iPinYou

Maintenant nous étudions la distribution de chaque attribut dans la base de données d'iPinYou préparée dans la section précédente afin de générer une nouvelle base de données ajustable. Nous utilisons le logiciel JMP [3] développé par SaaS permettant de comparer les données de chaque attribut avec des distributions théoriques discrètes et continues tel que Gamma, Exponentielle, etc., et les compare selon les critères suivantes :

[3] https://www.jmp.com/en_us/download-jmp-free-trial.html

- **AICc (Akaike corrigé) :** mesure de sélection et de comparaison des modèles basée sur la valeur -2 log de vraisemblance L et le nombre d'échantillons k. Le critère d'information d'Akaike s'écrit comme suit :

  $AIC = 2k - 2\ln(L)$

  L'AICc est une correction de l'AIC dans le cas d'échantillons de petite taille :

  $AICc = AIC + \frac{2k(k+1)}{n-k-1}$

  On choisit alors le modèle avec le critère d'information d'AICc le plus faible.

- **BIC (Bayésien) :** mesure de sélection et de comparaison des modèles basée sur la valeur -2 log de vraisemblance et le nombre d'échantillons k aussi. Les petites valeurs indiquent de meilleurs modèles.

  $BIC = -2\ln(L) + \ln(n)k$

- **Courbe de fonction de répartition :** est la courbe de la fonction FX qui, à tout réel x, associe la probabilité d'obtenir une valeur inférieure ou égale :

  $F_X(x) = \mathbb{P}(X \leq x)$

  On choisit le modèle avec la courbe la plus proche de celle de la distribution de nos données

- **QQ plot et PP plot :** permettent d'évaluer la pertinence de l'ajustement d'une distribution donnée à un modèle théorique.

Maintenant, nous exécutons le logiciel JMP sur chaque attribut de la base de données préparée. Pour (RegionID, CityID) et (AdSlotWidth, AdSlotHeight), nous les examinons ensemble, car ils sont corrélés.

Nous obtenons pour chaque attribut et ses ensembles de données sa distribution théorique la plus proche ainsi que les détails des critères de comparaison vue précédemment :

- **Timestamp :** distribution continue normal d'ordre 3 (figure 4.4).
- **Type du log :** distribution discrète constante
- **Emplacement (RegionID, CityID) :** distribution discrète bêta binomial avec moyenne = 66 et écart-type = 77 (figure 4.5).

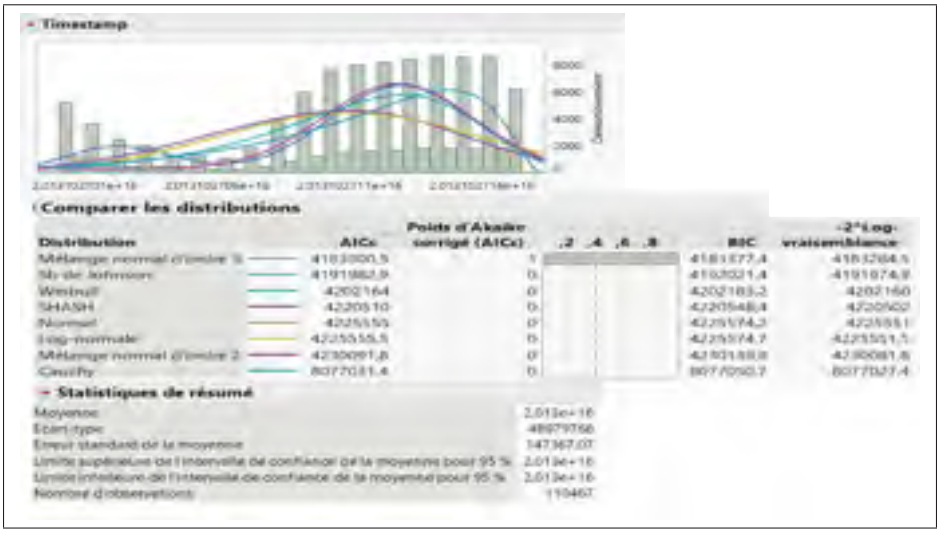


Figure 4.4 Résultats de JMP pour l'attribut Timestamp

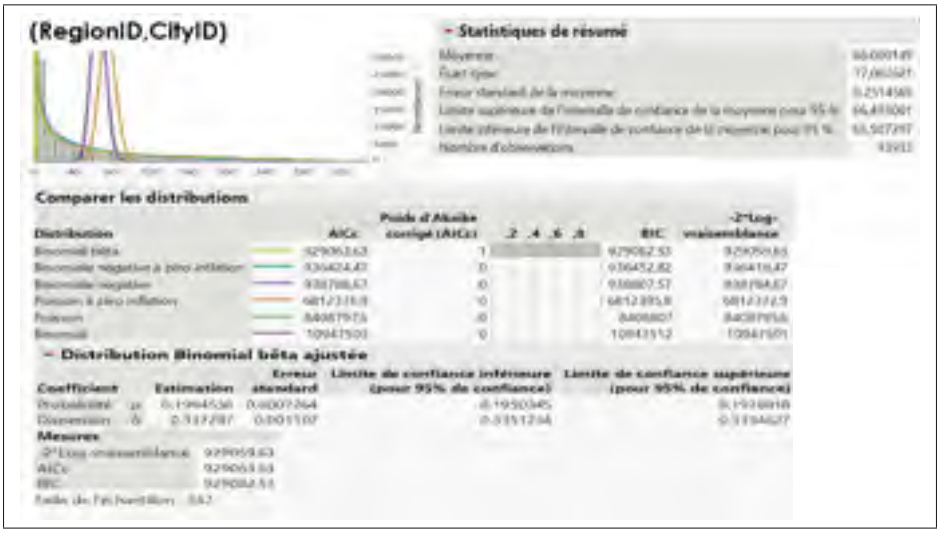


Figure 4.5 Résultats de JMP pour l'emplacement d'utilisateur

- **Ad Exchange :** distribution discrète binomiale bêta avec moyenne = 2,2388 et écart-type = 0,7821 (figure 4.6).
- **Taille de l'espace publicitaire (AdSlotWidth, AdSlotHeight) :** distribution discrète binomiale négative avec moyenne = 3,527 et écart-type = 2,329 (figure 4.7).

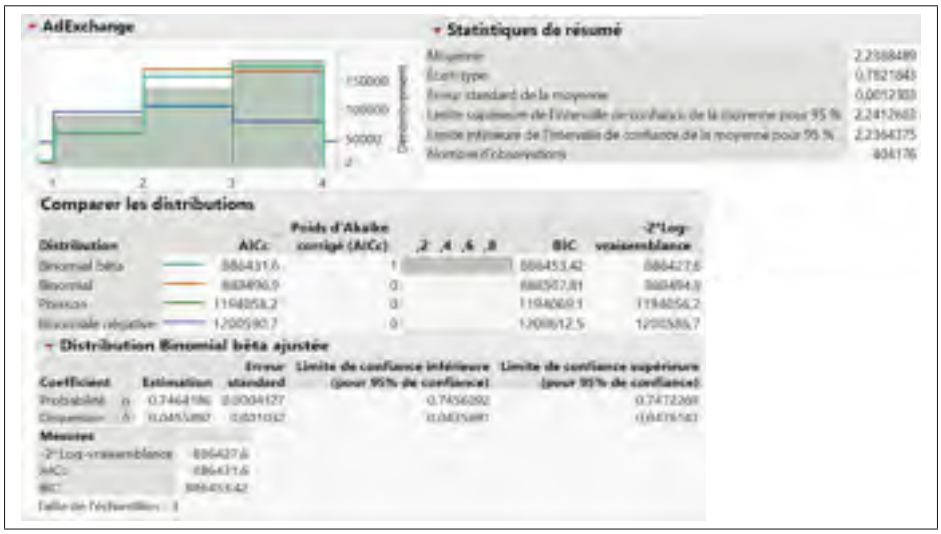


Figure 4.6 Résultats de JMP pour Ad Exchange

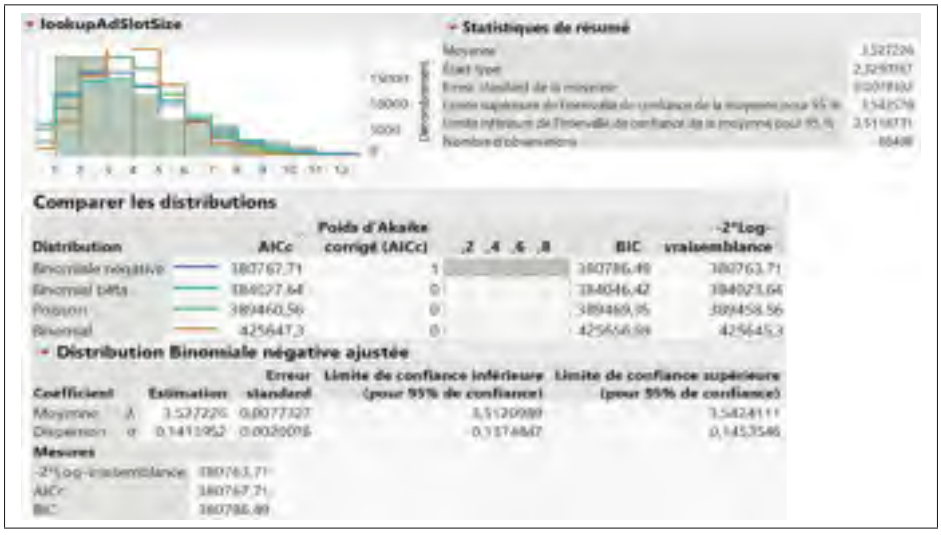


Figure 4.7 Résultats de JMP pour taille d'espace d'annonce

- **Visibilité de l'espace publicitaire :** distribution discrète binomiale bêta avec moyenne = 0,867 et écart-type = 0,939 (figure 4.8).
- **Prix de réserve :** distribution discrète binomiale négative à inflation zéro avec moyenne = 28,977 et écart-type = 43,709 (figure 4.9).
- **AdvertiserID :** distribution discrète constante

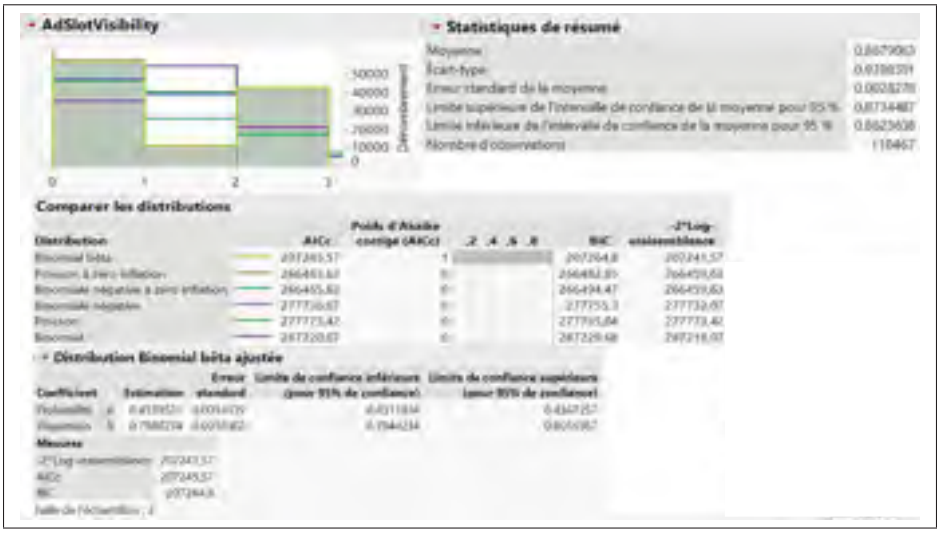


Figure 4.8 Résultats de JMP pour visibilité d'espace d'annonce

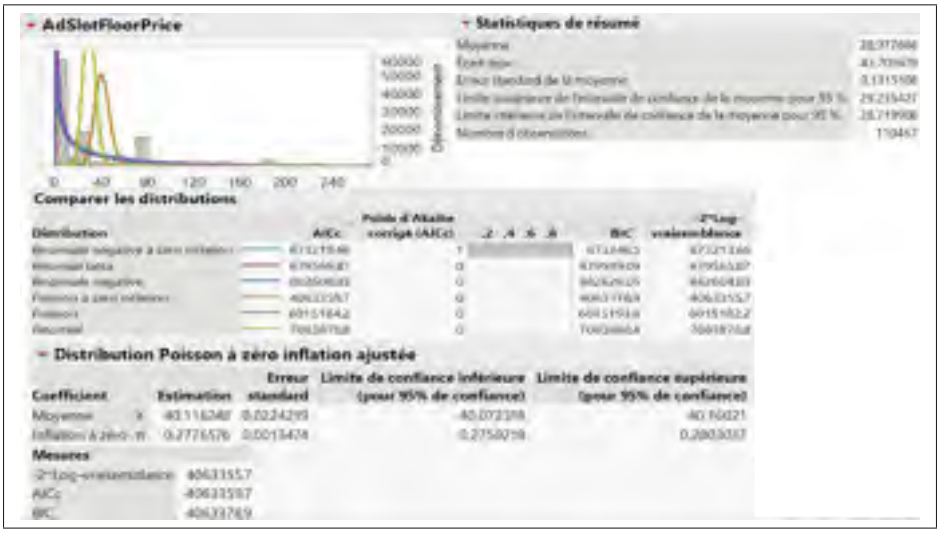


Figure 4.9 Résultats de JMP pour le prix de réserve

- **IDs des profiles d'utilisateur :** distribution discrète binomiale négative avec moyenne = 10804,419 et écart-type = 1622,4653 (figure 4.10).

Enfin, nous implémentons notre générateur de requêtes ajustable en générant pour chaque attribut des valeurs aléatoires ayant les mêmes distributions de la base de données d'iPinYou, mais avec des paramètres personnalisés selon le besoin dans l'évaluation (chapitre 6).

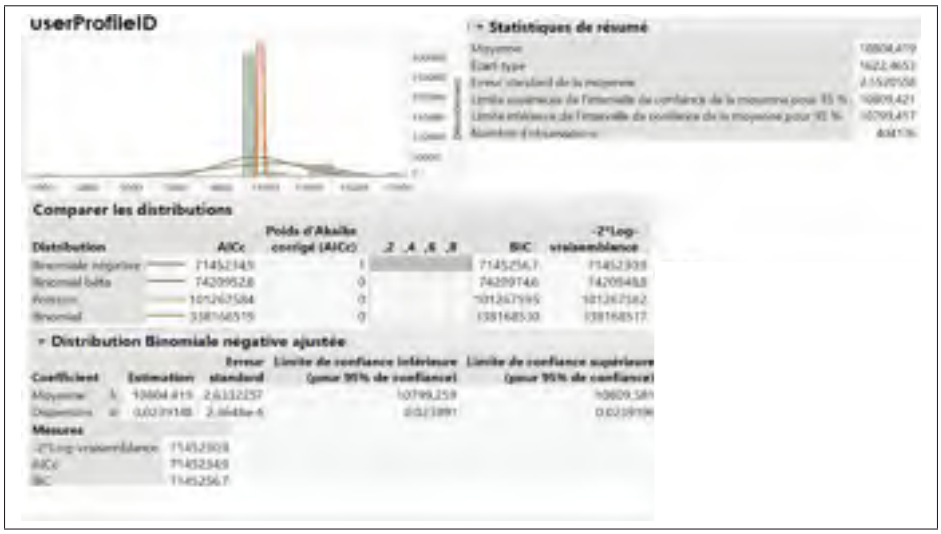


Figure 4.10 Résultats de JMP pour IDs des utilisateurs

4.4 Conclusion

Dans ce chapitre, nous avons présenté les détails de l'implémentation de notre projet de recherche qui consiste à l'intégration de Pub/Sub avec filtrage Top-k dans OpenRTB. Puis, nous avons introduit un démo qui visualise les résultats des enchères et la comparaison des trois implémentations. Enfin, nous avons proposé un générateur issu de la base de données d'iPinYou permettant d'engendrer des requêtes d'enchères ajustables pour l'utiliser dans l'évaluation.

CHAPITRE 5

EXPÉRIENCE PERSONNELLE

Le but principal de notre recherche est l'intégration du modèle Publish/Subscribe avec Top-k dans Real-Time Bidding. Dans ce cadre, nous avons lu plusieurs articles liés à notre sujet et qui s'intéressent au système Publish/Subscribe et ses différents fonctionnalités expressives tels que l'agrégation, l'abonnement évolutif et le filtrage Top-k ainsi que Real-Time Bidding pour comprendre ses concepts et son fonctionnement.

Ensuite, nous avons commencé à chercher une implémentation de RTB, mais malheureusement nous n'avons pas trouvé beaucoup de projets open source. La seule implémentation disponible est celle de OpenRTB, mais elle ne contient pas l'implémentation d'AdX qui est un composant principal dans RTB où nous allons intégrer Pub/Sub. Alors, nous avons implémenté un AdX selon la spécification décrite dans OpenRTB. En outre, le DSP implémenté dans OpenRTB est de type simple comme nous avons vu dans la section 4.1 donc nous avons cherché d'autres implémentations open source de DSP, mais nous n'avons pas trouvé, et par conséquent nous avons développé deux autres types basiques qui sont Random et Below_eCPC.

Puis, nous avons dû choisir l'implémentation de Pub/Sub la plus convenable pour l'intégrer dans OpenRTB qui doit être au premier lieu basé sur le contenu pour exprimer les intérêts des DSPS. Après avoir étudié les implémentations existantes, nous avons choisi Header exchange de RabbitMQ, qui non seulement efficace dans notre cas, mais aussi caractérisé par une documentation claire [1] et une implémentation complète.

Après avoir choisi l'implémentation de RTB et de Pub/Sub, nous avons eu besoin d'une base de données pour tester nos solutions. Selon notre connaissance, la seule base de données de RTB réelle et publique disponible est iPinYou dataset [2]. Donc il y a un manque de base de données de test dans le domaine de RTB.

[1] https://www.rabbitmq.com/tutorials/tutorial-three-java.html

[2] https://www.dropbox.com/s/txz0ms0axqf7jrl/ipinyou.contest.dataset.7z?dl=0

Pour commencer le développement, nous avons installé RabbitMQ ainsi que Erlang indispensable pour son fonctionnement, en suivant le guide d'installation disponible dans son site web officiel.

Ensuite, nous avons développé nos solutions avec Java/J2EE en utilisant l'IDE Eclipse. Cet IDE est massivement utilisé en entreprise et un outil puissant, gratuit, libre et multiplateforme. Pour exécuter nos projets, nous avons eu besoin de mettre en place un serveur d'applications.Nous avons choisi au début d'utiliser Tomcat, car c'est un serveur léger, gratuit, libre, multiplateforme.

En exécutant nos solutions avec iPinYou dataset, nous avons constaté que le serveur Tomcat ne peut pas exécuter toute la base de données voire qu'il a un temps maximal alloué. Nous avons essayé d'augmenter ce temps, mais ça n'a pas marché. Donc nous avons changé Tomcat par Glassfish qui est plus puissant puisque ce problème est disparu après. Nous avons également eu d'autres erreurs d'exécution et des exceptions, mais nous les avons résolus grâce aux forums surtout Stackoverflow.

Nous avons testé avec iPinYou dataset jusqu'à que nous avions l'idée de créer un générateur de requêtes d'enchères ajustables issu de la base de données de départ, en créant de nouvelles données avec les mêmes distributions de leurs attributs. Donc, d'abord nous avons eu besoin de savoir les distributions des attributs de iPinYou dataset, qui sont presque toutes discrètes. Or en utilisant Python, nous avons constaté que ses frameworks sont capables d'ajuster seulement des données continues aux distributions théoriques. Alors, nous avons cherché d'autres outils qui extraient la distribution des données discrètes, mais nous n'avons trouvé que des logiciels non gratuits parmi eux JMP qui fournit une version d'essai, donc nous l'avons utilisé pour extraire toutes les distributions de iPinYou dataset.

En conclusion, nous avons résumé les étapes et les problèmes rencontrés lors de la réalisation de notre projet de recherche. Cela a été une expérience intéressante pour nous et elle nous a beaucoup appris de nouvelles connaissances.

CHAPITRE 6

ÉVALUATION

Dans cette section, nous évaluons expérimentalement nos solutions : Pub/Sub et Top-k par rapport à l'implémentation de base OpenRTB. Nos expériences contiennent une analyse de performance des trois approches, nous les comparons en termes de nombre des messages échangés, de temps d'enchère, de prix payé et d'efficacité. Ensuite, nous effectuons une analyse de sensibilité des principaux paramètres impactant les solutions.

6.1 Configuration expérimentale

Nous effectuons notre évaluation en utilisant une machine virtuelle Ubuntu version *18.04.3* LTS, avec *4096 Mo* de RAM et *2 GHz* CPU. Par défaut, nous utilisons la base de données générée, vue dans la section 4.3 et issue de la base de données iPinYou avec des paramètres ajustables. Nous choisissons de générer *100* requêtes d'enchères. Dans nos expériences, nous définissons la sélectivité de *70 %* des DSPs intéressés par la requête d'enchère, et nous utilisons par défaut la méthode Top-k basée sur le score (section 3.3), avec $k = 10$.

6.2 Comparaison des performances

Nous comparons nos trois approches : OpenRTB (approche de base), Pub/Sub uniquement et Pub/Sub avec Top-k en termes de nombre de messages, temps de traitement, prix d'enchère et efficacité.

6.2.1 Nombre de messages

La figure 6.1 montre le nombre de réponses aux enchères envoyées par les DSPs ainsi que le taux de réponse vide ou sans offres. Par rapport à l'approche de base, Pub/Sub et Top-k réduisent le nombre total de messages de *47 %* et *90 %*, respectivement. En particulier, *4696* de réponses vides sont reçus par l'AdX, qui sont complètement éliminées par nos deux solutions. La solution

Top-k peut filtrer davantage un nombre de messages supplémentaires de *4304* par rapport au Pub/Sub seul, qui sont des réponses aux enchères non compétitives. De plus, le filtrage Top-k augmente la stabilité, car le DSP reçoit *10* réponses exactement sont reçues pour chaque requête d'enchère. En diminuant les réponses aux enchères, le système RTB sera optimisé en terme de bande passante puisque les réponses sont envoyés à l'aide du protocole HTTP ainsi qu'ils sont au format complexe JSON.

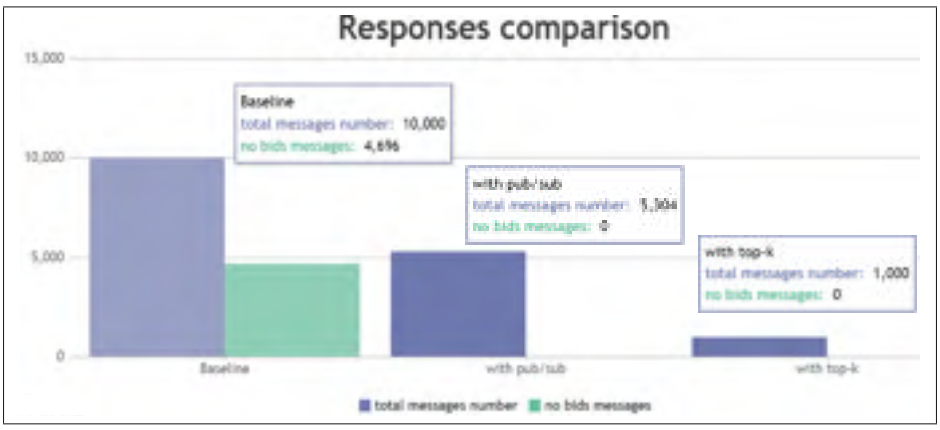


Figure 6.1 Comparaison de nombre de réponses aux enchères

6.2.2 Temps d'enchère

La figure 6.2 montre la fonction de distribution cumulative (CDF) pour le temps d'enchère de bout en bout dans les trois approches. Le temps d'enchère est mesuré au moment où l'AdX reçoit la demande d'enchère de l'éditeur jusqu'au moment où il contacte le DSP gagnant. Dans l'approche de base OpenRTB, les enchères prennent entre *1 800 ms* et *3 000 ms*, tandis que *800-2 000 ms* pour Pub/Sub et *120-600 ms* pour Top-k. La médiane de chaque solution est de *2510 ms*, *1171 ms* et *140 ms* pour l'approche de base OpenRTB, Pub/Sub et Top-k respectivement. Nos solutions Pub/Sub et Top-k sont donc en mesure de réduire les délais d'enchères de *50 %* et *94 %* respectivement. Ce résultat démontre qu'une réduction du nombre de DSPs contacté par l'AdX diminue les chances de rencontrer un DSP lent (retardateur), réduisant ainsi le temps global.

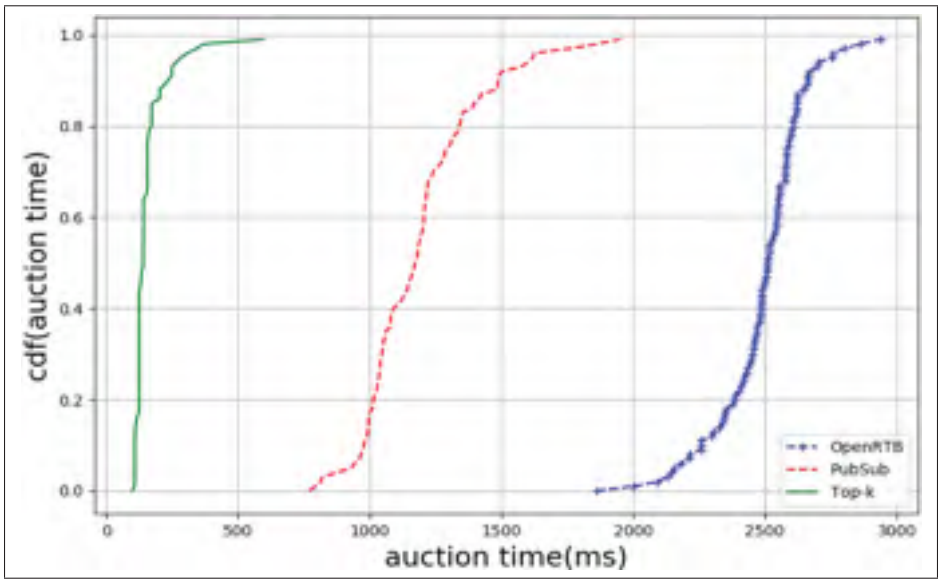


Figure 6.2 Comparaison du temps des enchères

6.2.3 Prix payé

La figure 6.3 montre le CDF pour le prix payé en utilisant les mêmes requêtes d'enchères pour les trois approches. Ici, l'approche de base OpenRTB est optimale, car elle contacte tous les DSPs à chaque fois, garantissant le prix maximum de chaque enchère. La solution Pub/Sub fonctionne également de manière optimale, car elle élimine uniquement les réponses vides qui n'affecte pas l'enchère. Pour la solution Top-k, une perte médiane de *3,75 %* est observée pour deux raisons. Tout d'abord, un DSP avec un prix d'offre élevé mais un temps de réponse lent peut ne pas être sélectionné, car il obtiendra un mauvais score dans la sélection Top-k. Deuxièmement, le modèle de prédiction utilisé pourrait sous-estimer de manière incorrecte le score d'un DSP, ce qui entraînerait qu'il ne soit pas sélectionné alors qu'il aurait eu un impact sur le prix payé (c'est-à-dire qu'il a soumis une offre supérieure au deuxième meilleur prix).

6.2.4 Efficacité

La figure 6.4 évalue l'efficacité des trois solutions. Cette métrique combine les deux métriques précédentes en divisant le prix payé par le temps de l'enchère. Étant donné que la solution

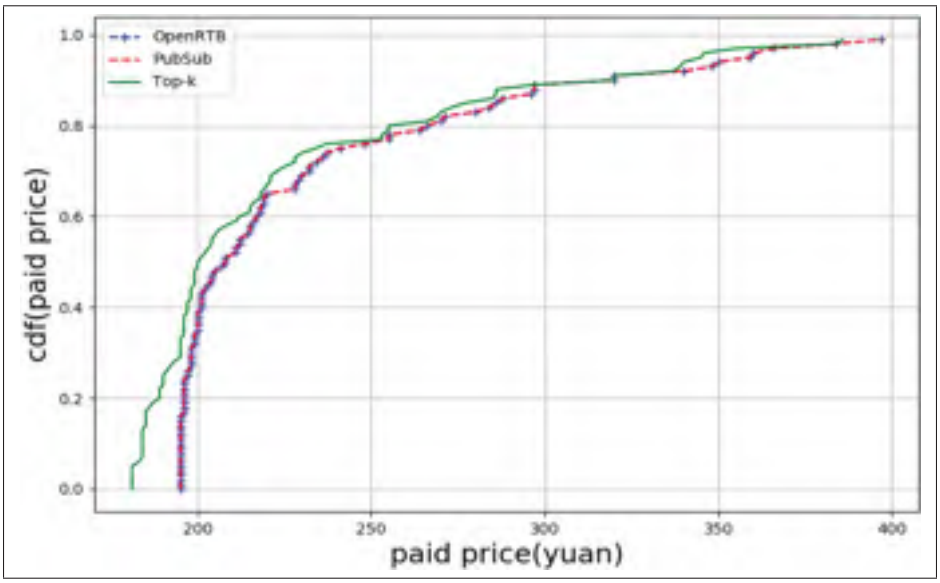


Figure 6.3 Comparaison des prix payés

Pub/Sub donne les mêmes prix payés qu'OpenRTB avec des délais d'enchères plus courts, son efficacité est supérieure avec une médiane de $\simeq$ *0,19 yuan/ms* par rapport seulement $\simeq$ *0,08 yuan/ms*, donc *2,375* fois. Cependant, Top-k est encore meilleur avec une efficacité médiane de $\simeq$ *1,48 yuan/ms*, ce qui est *18,5* fois meilleure que l'approche de base et *7,79* fois mieux que la solution pub/sub seul. Bien que Top-k sacrifie une petite perte de prix payés, son temps d'exécution est également considérablement plus court. Le compromis est donc en faveur de la solution Top-k.

Résumé

Par rapport à l'approche de base, les approches Pub/Sub et Top-k réduisent le nombre de messages de *47 %* et *90 %*, et le temps des enchères de *50 %* et *94 %*, respectivement. L'approche Pub/Sub seul n'affecte pas le prix payé, tandis que l'approche Top-k résulte une perte de *3,75 %* des prix payés. Cependant, cette perte est compensée par une accélération considérable du temps des enchères et une efficacité supérieure. Donc l'approche Top-k est souhaitable si l'application reçoit un flux quasi infini de requêtes qui doivent être traitées efficacement. En revanche, si le volume des requêtes d'enchères est limité, l'approche pub/sub seul est souhaitable afin d'obtenir le prix maximum par enchère.

Figure 6.4 Comparaison d'efficacités des DSPs

6.3 Analyse de sensibilité

Maintenant, nous effectuons une analyse de sensibilité de nos solutions afin d'étudier l'impact de trois paramètres : sélectivité des requêtes des enchères, valeur de k, et choix du modèle de prédiction.

6.3.1 Sélectivité des DSPs

La figure 6.5 montre l'impact de la variation de la sélectivité des DSPs. Ce paramètre contrôle le nombre de DSPs intéressés par chaque requête d'enchère. Nous testons 3 scénarios : *30 %*, *50 %* et *70 %*. L'approche de base n'est pas affectée par ce paramètre, car elle contacte toujours tous les DSP sans tenir compte de leurs intérêts. La solution Top-k aussi est généralement insensible à la sélectivité : si k est suffisamment faible, le changement de sensibilité n'a pas d'impact sur le nombre de DSPs sélectionnés, qui est toujours k par requête d'enchère. Par contre, l'approche Pub/Sub seul est sensible à la sélectivité, ses médianes sont *999 ms*, *1083 ms* et *1171 ms* pour les scénarios *30 %*, *50 %* et *70 %* respectivement. En d'autres mots, tant que la sélectivité des DSPs augmente, le filtrage effectué par le Pub/Sub diminue, ce qui réduit l'efficacité de la solution.

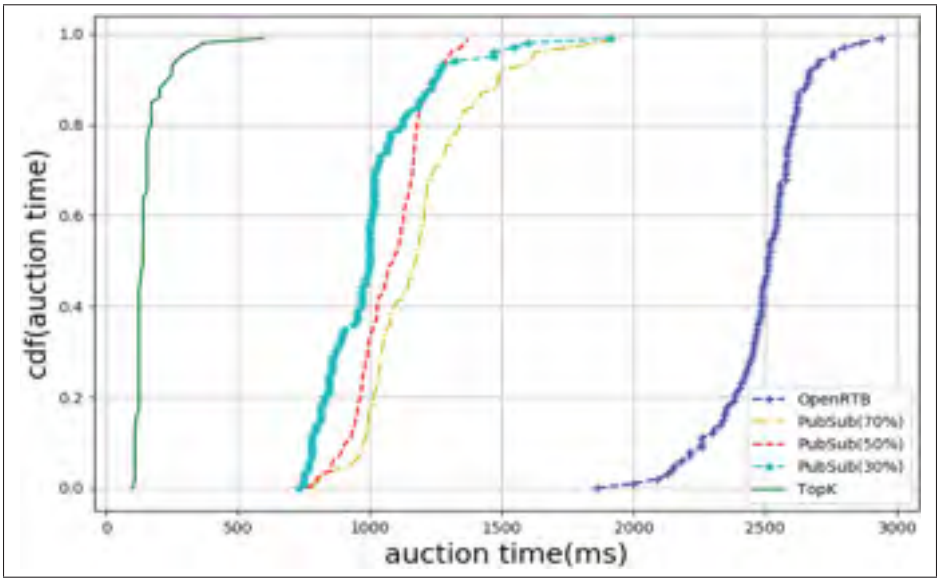


Figure 6.5 Sélectivité des DSPs

6.3.2 Paramètre k

Nous testons 4 différentes valeurs de k : *2, 4, 8* et *10*. La figure 6.6 montre le CDF pour les prix payés pour chaque scénario, par rapport au prix optimal représenté par l'approche de base OpenRTB. Comme prévu, le prix payé diminue généralement lorsque k diminue, car la marge d'erreur devient plus étroite pour que le modèle de prédiction identifie correctement les DSPs idéaux pour l'enchère. Cependant, la différence n'est pas substantielle, avec une perte médiane de *4,1 %* pour *k=2* comparativement à *3,61 %* pour *k=10*. Cela indique que le modèle de régression utilisé est très précis.

Pour le temps des enchères, la figure 6.7 montre qu'une amélioration notable peut être obtenue en réduisant la valeur de k. La valeur médiane de *k=2* est de *32 ms* par rapport à *140 ms* pour *k=10*

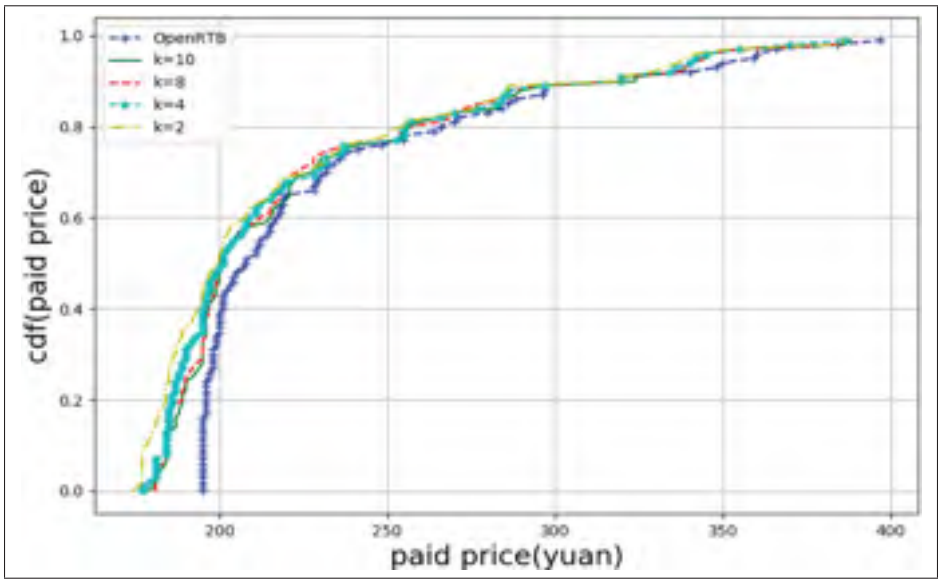


Figure 6.6 Paramètre *k*- prix payés

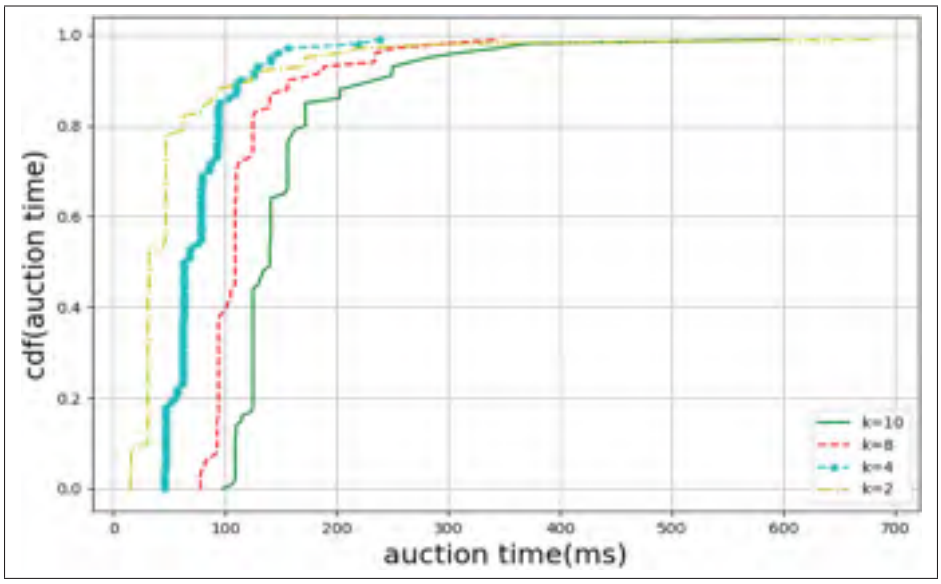


Figure 6.7 Paramètre *k*- temps des enchères

6.3.3 Modèles de Top-k

Nous comparons nos trois modèles de Top-k : basé sur le score, basé sur le rang et basé sur la sélection binaire, par rapport à deux approches de base : basée sur l'historique (moyenne des enchères passées) et aléatoire (choisir k DSPs au hasard).

La figure 6.8 montre le CDF pour les prix payés. Les deux approches de base : historique et aléatoire sont clairement inférieures à notre solution proposée, avec des intervalles de *120-340 yuan* et *100-310 yuan* respectivement. Une exception notable est celle de Top-k basé sur le rang, qui a obtenu de mauvais prix pour *20 %* des enchères et a donc un large intervalle de prix *100-390 yuan*. D'un autre côté, les prix payés dans Top-k basés sur le score et la sélection binaire sont de l'ordre de *180-390 yuan*, ce qui est presque optimal par rapport à l'approche de base OpenRTB.

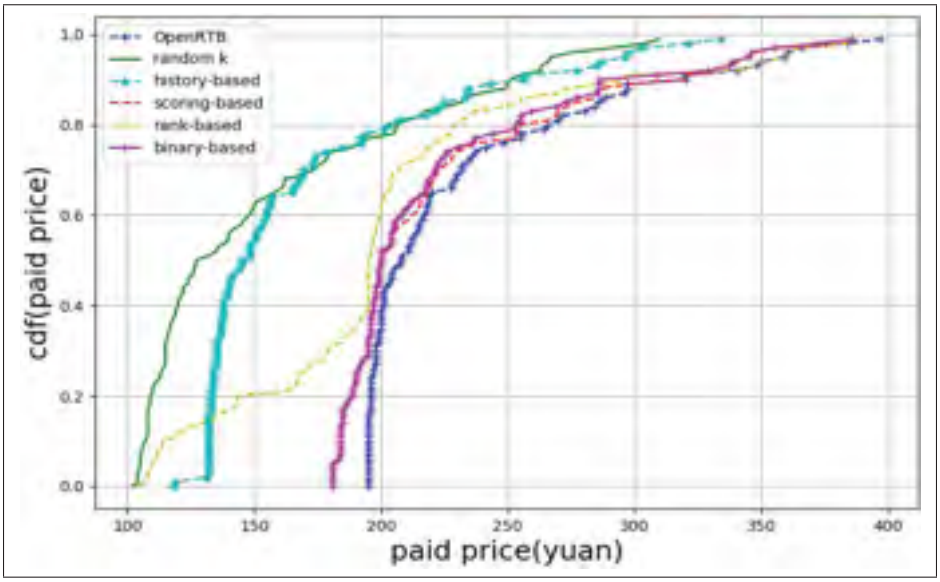


Figure 6.8 Comparaison des modèles de Top-k - prix payés

La figure 6.9 montre le CDF pour le temps aux enchères. Les approches Top-k aléatoire et historique ont un intervalle de *40-300 ms* et *70-450 ms*. Par contre, le modèle basé sur le score prend plus de temps de calcul, avec un intervalle de *100-600 ms*. Le modèle basé sur le rang a

un intervalle de *40-330 ms* et celui basé sur la sélection binaire surpasse tous les autres avec le meilleur intervalle de *40-220 ms*.

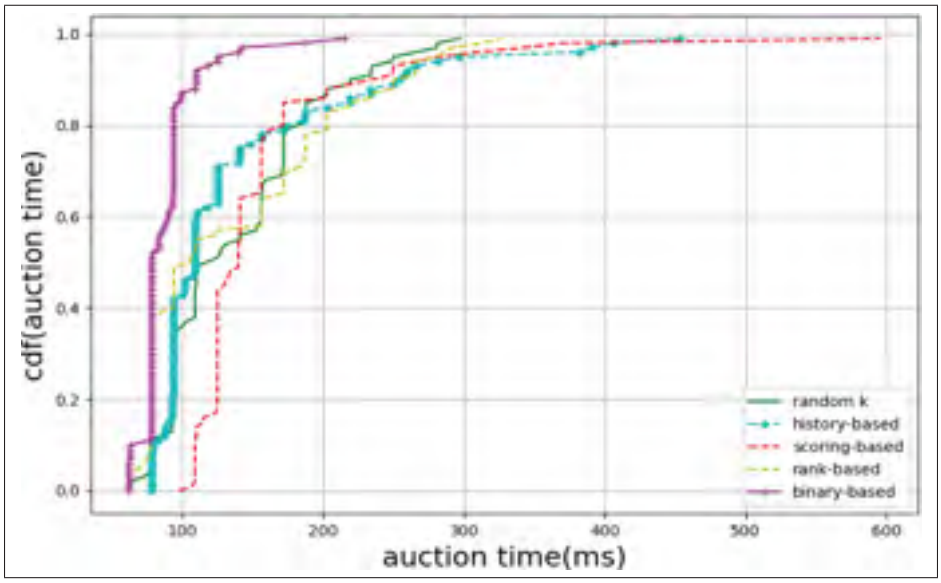


Figure 6.9 Comparaison des modèles de Top-k - temps des enchères

Nous comparons également l'efficacité (prix payé/temps d'enchère) de la figure 6.10 qui montre clairement que le modèle basé sur la sélection binaire est le plus efficace avec un intervalle de *0,9-5 yuan/ms* par rapport aux autres modèles qui sont presque dans le même intervalle *0,5-4 yuan/ms*.

Nous mesurons la variation du nombre k des DSPs dans la figure 6.11. Nous notons que le modèle basé sur la sélection binaire ne garantit pas un nombre k fixe pour chaque requête d'enchère, il varie entre *3* et *11* DSPs sélectionnés, puisque le système prend ceux ayant les réponses "oui" sans tenir compte de leur nombre. Pour les modèles basés sur le rang et le score, les résultats confirment que le nombre de DSPs sélectionnés est toujours égal à k (*10*).

Résumé

La solution Top-k se considère comme étant la plus fiable par rapport à l'approche de base OpenRTB et Pub/Sub seul, car elle n'est pas sensible à la sélectivité des DSPs. Mais, ses

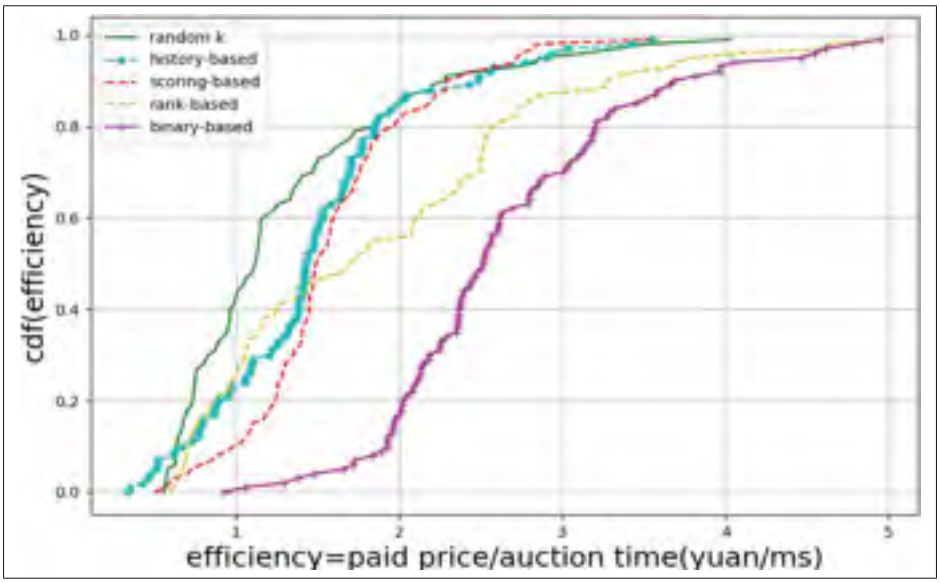


Figure 6.10 Comparaison des modèles de Top-k - efficacité

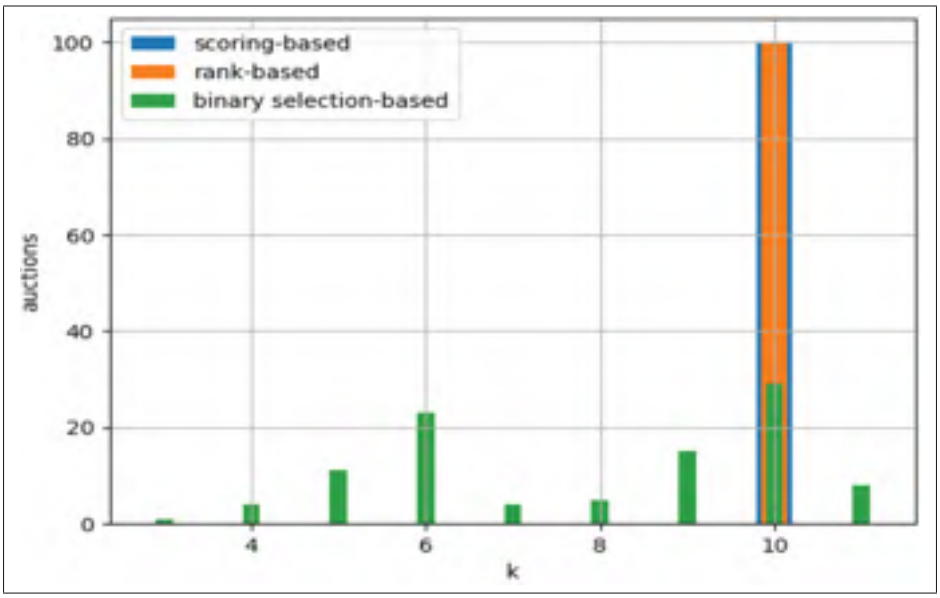


Figure 6.11 Nombre des DSPs sélectionnées dans les modèles de Top-k

résultats peuvent changer en variant le paramètre *k* en termes de prix payé et de temps d'enchère. Nous comparons également les modèles de Top-k dans le tableau 6.1. Le modèle basé sur la sélection binaire montre une efficacité maximale puisqu'il est le plus rapide avec une perte de

prix minimale, mais avec un nombre k de DSPs non fixe. Le modèle basé sur le score a la pire efficacité, mais peut obtenir de meilleurs prix que d'autres solutions. L'approche basée sur le rang est un compromis entre les deux, car elle a une efficacité moyenne avec un nombre k fixe.

Tableau 6.1 Comparaison des modèles Top-k

modèles du Top-k	**avantages**	**inconvénients**
basé sur le score	prix payés proches de ceux de OpenRTB	prédiction compliquée des variables continues
basé sur le rang	modèle simple	prix payés ne sont pas toujours proches de ceux de OpenRTB
basé sur la sélection binaire	meilleur performance	le paramètre k n'est pas stable et n'est pas prévisible, donc c'est difficile de savoir en avance combien de ressources nous avons besoin.

CONCLUSION ET RECOMMANDATIONS

Dans la spécification 2.5 de RTB, AdX diffuse les requêtes des enchères à tous les DSPs, ce qui génère un surcoût de communication et de calcul massifs. Nous avons proposé d'utiliser le modèle Pub/Sub pour exprimer les intérêts des DSPs sous forme d'abonnements basés sur le contenu, afin d'éliminer les réponses vides.

De plus, nous avons réduit davantage le nombre de DSPs contactés en diminuant les offres non compétitives ou ayant des temps de réponse lents. Nous avons formulé le problème du nombre optimal de réponses aux enchères et montrons comment le filtrage Top-k peut être utilisé pour résoudre ce problème dans le cadre réel. Nous avons également proposé trois types de filtrage Top-k : le premier est basé sur le calcul de score et repose sur l'analyse de régression avec des variables continues, afin d'estimer quels DSPs devraient participer à une telle enchère, tandis que les deux autres modèles basés sur le rang et la sélection binaire utilisent une analyse discrète.

Notre évaluation a confirmé l'efficacité de nos approches pour réduire le nombre de messages (en éliminant toutes les réponses vides) et le temps des enchères, tout en maintenant des prix gagnants optimaux ou quasi optimaux (dans le cas de Top-k).

En somme, la solution la plus efficace est OpenRTB avec Pub/Sub basé sur le contenu avec un filtrage Top-k basé sur la sélection binaire.

Pour nos travaux futurs, nous souhaitons tester l'applicabilité de notre approche avec d'autres de bases de données au-delà d'iPinYou, et mettre en œuvre des stratégies d'enchères plus sophistiquées qui pourraient contester la précision de nos modèles de Top-k. Nous chercherons également à rendre notre solution autoadaptative, en ajustant dynamiquement ses paramètres (par exemple, la valeur de k) selon les conditions actuelles.

ANNEXE I

ARTICLE SOUMIS

Cet article, intitulé "Selective Auctioning using Publish/Subscribe for Real-Time Bidding", est soumis à DEBS 2020, 14TH ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED AND EVENT-BASED SYSTEMS (July 2020, Montréal).

Selective Auctioning using Publish/Subscribe for Real-Time Bidding

Authors anonymized

ABSTRACT

Real-Time Bidding (RTB) advertising has recently experienced a massive growth in the industry of online marketing. RTB technologies allow an Ad Exchange (AdX) to conduct online auctions in order to sell targeted ad impressions by soliciting bids from potential buyers, called Demand Side Platforms (DSPs). In the OpenRTB specifications, which is a well-known open standard protocol for RTB, the AdX sends bid requests to all DSPs for every auction. This communication protocol is highly inefficient since for each given auction, only a small fraction of DSPs will actually submit a competitive bid to the AdX. The exchange of bid requests to uninterested parties waste valuable computation and communication resources. In this paper, we propose to leverage publish/subscribe to optimize the auction protocol used in RTB. In particular, we demonstrate how RTB semantics can be expressed using content-based subscriptions, which allows for selective dissemination of bid requests in order to eliminate no-bid responses. We also formulate the problem of minimizing the number of bid responses per auction, and propose combining top-k scoring with regression analysis with continuous variables as a heuristic solution to further reduce the number of irrelevant responses. We then adapt our solution by considering discrete machine learning models for a faster execution. Finally, we evaluate our proposed solutions against the OpenRTB baseline in terms of end-to-end latency and total paid price over time efficiency.

CCS CONCEPTS

• **Information systems → Online advertising**; • **Computer systems organization → Distributed architectures**; • **Software and its engineering → Publish-subscribe / event-based architectures**.

KEYWORDS

Real-time bidding, online advertising, content-based publish/subscribe system, top-k filtering, machine learning

ACM Reference Format:

Authors anonymized. 2020. Selective Auctioning using Publish/Subscribe for Real-Time Bidding. In *14th ACM International Conference on Distributed and Event-based Systems, July 13–17, 2020, Montreal, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DEBS '20, July 13–17, 2020, Montreal, Canada

© 2020 Association for Computing Machinery.

ACM ISBN XXX-X-XXXX-XXXX-X/XX/XX…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Real-Time Bidding (RTB) is a type of online advertising that allows websites to sell in real-time an ad impression to the highest bidder. RTB is a form of targeted advertising as each advertiser calculates its bid based on certain characteristics such as the banner size, the context of the web page, the user profile, etc.

OpenRTB is a specification of RTB system [6] which proposes open industry standards for communication between buyers and sellers of ad impressions. It is based on two main components: Ad Exchange (AdX) and Demand Side Platform (DSP). AdX is an intermediate agent between sellers and buyers of ad impressions. The AdX runs an auction for each ad request and sends corresponding bid requests to all eligible DSPs. Each DSP serves multiple advertisers, which can ask the DSP to run an ad campaign for a particular product based on target audience, predefined budget and campaign duration [13]. Upon receipt of a bid request by the AdX, each DSP calculates the bid price based on the ad campaigns of its advertisers. The AdX collects bid responses to the auction and selects the winning DSP accordingly.

In the current OpenRTB specifications, the AdX broadcasts every bid request to all DSPs and awaits their bid responses. However, not every bid request will be of interest to each DSP, which may reply with a no-bid response [6]. Furthermore, bid responses with non-competitive bid amounts have no realistic chance of winning an auction. In light of these observations, we conclude that OpenRTB wastes significant resources due to the sending of irrelevant bid requests and responses.

The impact of this overhead is characterized by two factors. First, bid request and bid response data are exchanged using HTTP in JSON [6], which is a heavy human-readable format. Second, a timeout value (called t_{max}) is fixed for each auction by the publisher, which is set to a sufficiently high value to collect all bid responses, including no-bid responses. Most of publishers have restriction of getting an Ad within t_{max} between 100 ms and 300 ms [9].

In this paper, we propose a solution to reduce the communication overhead of RTB by selecting the subset of DSPs to be contacted for each bid request. We integrate Publish/Subscribe (Pub/Sub) semantics in the OpenRTB implementation to allow DSPs to express their interests as content-based subscriptions [5]. The AdX can then act as a pub/sub broker and filter DSPs based on their subscriptions, thus eliminating no-bid responses.

Furthermore, we can reduce the overhead by eliminating non-competitive bids from each auction. We propose an extension to our pub/sub system which uses top-k filtering and regression analysis (with continuous variables) to select a subset of DSPs who are likely to submit competitive bids for a given auction. We then adapt our model in order to substitute continuous regression model with an efficient discrete one.

In this paper, we provide the following contributions:

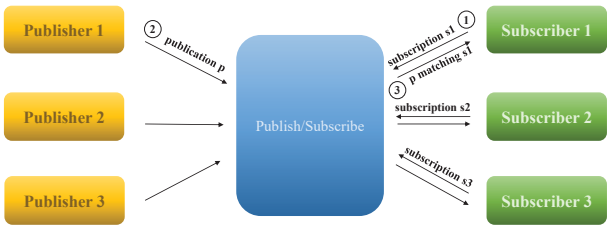


Figure 1: Publish/Subscribe Overview

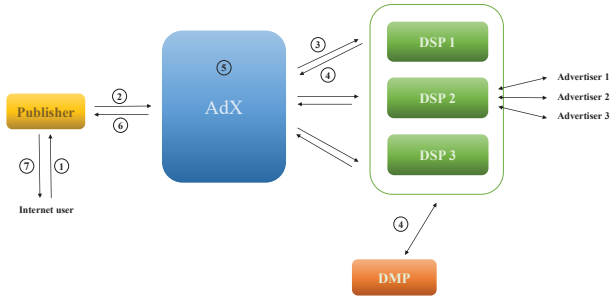


Figure 2: RTB Process Overview

- We propose a pub/sub solution which maps RTB semantics, where bid requests are modeled as publications and DSP interests are expressed as content-based subscriptions (Section 4.1),
- We formulate the problem of minimum bid responses selection for a RTB auction (Section 4.2),
- We leverage top-k filtering on top of pub/sub to further filter DSPs based on predicted bid prices and responses time, which are computed using regression analysis (Section 4.3),
- We adapt our solution in order to employ discrete machine learning models, thereby improving the speed of the top-k filtering (Section 5) and finally,
- We implement our approach by integrating the OpenRTB specifications with the RabbitMQ pub/sub system and evaluate our solutions using the iPinYou dataset against a known baseline (Section 6).

We continue this paper with Section 2 on related works. We then present the background on publish/subscribe and real-time bidding in Section 3.

2 RELATED WORKS

In this section, we survey related works in the literature, divided into four categories: bidding strategies, header bidding and top-k filtering.

2.1 Bidding Strategies

As of today, RTB research remains limited given the relative size of its market [20]. Most of the research in this area is directed towards the optimization of DSP algorithms to calculate bid price. In [18], the authors estimate the probability of ad clicks and conversions using linear regression models (logistic regression and Bayesian Probit regression) and nonlinear models (factorisation machines, gradient tree models, and deep learning), then use this metric to optimize bidding. In [10], the authors suggest an algorithm which select high quality impressions and adjust the bid price based on historical performance, while spreading the ad campaign budget optimally across time. The algorithm is based on the estimation of click-through rate (CTR) and action or conversion rate (AR). In [21], the authors integrates the concept of budget utilization efficiency in DSP bidding strategy, in order to win as many impressions as possible. Another method proposed for bidding strategies consists in predicting the winning bid price [19]. These approaches are complementary to our own work, as we seek to reduce the communication overhead of conducting auctions by optimizing how the AdX selects DSPs. Once selected, the DSPs may then leverage the aforementioned works to decide its own bid.

These previous works may potentially reduce the processing time required by each DSP, and thus allow the AdX to set a shorter timeout (t_{max}). Our work is complementary to these bidding strategies since it also selects a subset of DSPs to contact, thereby reducing communication overhead. In addition, we implement some of these bidding strategies in our evaluation in order to generate a realistic load for bid responses (cf. Section 6).

2.2 Header Bidding

Header bidding or pre-bidding is a technique for online advertising that allows publishers to offer an ad impression to several ad exchanges before launching a RTB auction. Each AdX proposes a price to the publisher, who chooses the highest offer and sends the ad impression to the winning AdX [15], who then internally forwards it to the winning DSP. This process helps the publishers dynamically choose the most suitable AdX for each ad impression instead of committing to reservation contracts with each of them [16].

Header bidding is complementary to our solution: In order to propose a price, an AdX must conduct an auction for its own DSPs. Thus, the AdX can employ our solution to dynamically select the set of DSPs to contact for such a RTB auction.

2.3 Top-K Filtering

Top-k filtering is a commonly used technique to effectively reduce the volume of publications to be disseminated by a pub/sub system while maintaining high data relevance. In [17], the authors maintain top-k tweets for each social story at a news website serving high volume of page views using a publish/subscribe system. The tweets are ordered based on a content/recency score function. Top-k subscriptions is defined in [1] to deliver a publication only to the k best ranked subscribers using rank-covering for large-scale applications. Top-k filtering is also used in [2] to reduce the amount of notifications sent in social networks. In [4], the authors propose a new solution to handle a large number of temporal spatial-keyword (TaSK) top-k requests on a geo-text object stream.

In our work, we use top-k as a method to select the most relevant subscribers for a given publication, which is close to the model of top-k subscriptions presented in [1]. To the best of our knowledge, top-k filtering was never used in the context of RTB, and our

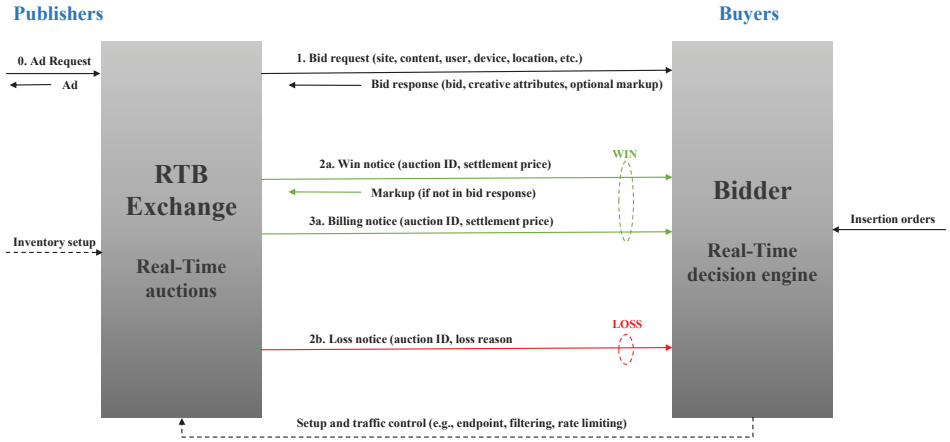


Figure 3: RTB Process in OpenRTB Specifications [6]

solution is adapted and optimized specifically according to RTB semantics.

3 BACKGROUND

In this section, we present different notions useful for understanding our work: Real-Time Bidding, OpenRTB specifications and Publish/Subscribe. We also describe the iPinYou dataset we use to conduct our evaluation.

3.1 Publish/Subscribe

Publish/Subscribe system allows data producers (publishers) to send publications which are delivered to matching data consumers (subscribers) according to their interests, expressed as subscriptions.

As Figure 1 shows, (1) each subscriber can express interest in a particular publication by sending a subscription to the intermediary service of Pub/Sub. (2) Once the publisher produces this publication, (3) it delivers it to its corresponding subscribers.

According [5], pub/sub system is characterized by scalability, a dynamic topology and decoupling in terms of space (publishers and subscribers do not know each other and do not know who sends or receives or even how many entities participating in the interaction), time (no need for interaction at the same time) and synchronization (it is asynchronous).

Subscription types commonly supported include topic-based, type-based and content-based [5]. In our work, we use content-based subscriptions, which are expressed as predicates over key-value pairs [3]. For example, the two following subscriptions Sub1 and Sub2 have two predicates each:

$$\text{Sub1} : (x = 3), (y > 4), \text{Sub2} : (x < 0), (y > 3)$$

A publisher publishes the following publication, which is only delivered to Sub1, since it satisfies all of its predicates:

$$\text{Pub} : (x = 3), (y = 5)$$

In Section 4.1, we use the aforementioned model to express DSP bidding interests as content-based subscriptions, while bid requests are modeled as content-based publications, carrying OpenRTB data as key-value pairs, which can be used to match against subscriptions by the pub/sub system.

3.2 Real-Time Bidding

Real-Time Bidding (RTB) is a form of online advertising which provides buying and selling of online ad impressions to a target audience, through real-time auctions that are conducted while the web page is loading (e.g., between 200 ms and 300 ms [9]).

As shown in Figure 2, the typical process [20] starts when advertiser asks a DSP to run an ad campaign for a particular product based on predefined budget, target audience and campaign duration. (1) Once a user visits a web page, (2) the AdX generates a bid request for each ad request and (3) sends it to all eligible DSPs, which takes into account their clients interests and user information provided by the Data Management Platform (DMP), and then calculates bid prices based on their campaigns, and sends bid response contains a bid price or no-bid reason for each bid request. (4) Then, the AdX collects bid responses received within a t_{max} specified by the publisher: all bid responses sent after this timeout are rejected. Finally, (5) the AdX determines the outcome of the auction and chooses the DSP with the best bid price to (6) display its ad to the publisher who (7) sends it to the user. The winner pays the second-highest price since RTB employs Vickrey auction semantics.

In our work, we focus on step (3) by modifying how the AdX behaves. Instead of broadcasting the bid request to all DSPs, it will performing a filtering action, using publish/subscribe semantics, to selectively decide which DSPs to solicit a bid response from.

3.3 OpenRTB Specifications

OpenRTB is a project developed by IAB Technology Laboratory to specify open communication protocols and standards between buyers and sellers of ad impressions in the context of RTB advertising [6]. OpenRTB provides an API with all essential entities (RTB Exchange and bidder) and interactions between them (sending bid request, bid response, win billing and loss notices), as shown in Figure 3. The RTB Exchange (AdX) sends a bid request to the bidder (DSP) with details about the site, content, user, device, location, etc. (cf. Table 4 in Appendix A), and the bidder returns bid response

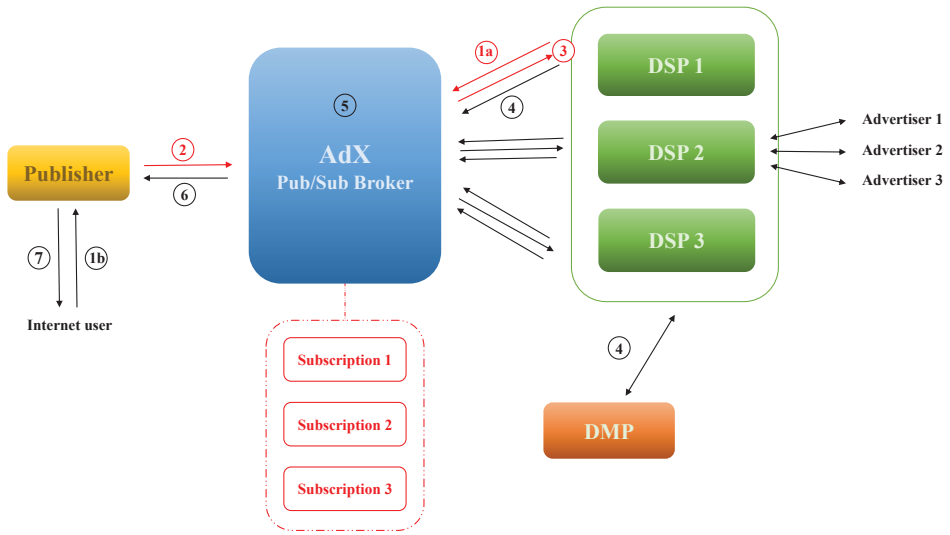


Figure 4: Proposed Content-Based Solution for Selective Auctioning

(cf. Table 5 in Appendix A) with a bid price if it decides to participate in the auction or no–bid if not. A possible response to the request is *no-bid*, which is accompanied by a reason (cf. Table 3 in Appendix A). After launching auction, AdX sends win notice to the winner and loss notice for other bidders.

In this paper, we focus on the possible reasons for no-bid responses. We analyse the different reasons and identify which ones could be avoided by integrating a publish/subscribe system to filter out DSPs using information gathered from the bid requests and bid responses.

3.4 iPinYou Dataset

iPinYou dataset is a set of auction, impression, click and conversion logs extracted from real advertising campaigns obtained from the iPinYou DSP platform. To the best of our knowledge, this is the only publicly available dataset containing RTB auctions. The original objective of the iPinYou dataset is to evaluate DSP bidding algorithms submitted for a competition in 2013 [11]. In this paper, we are only interested in the auction logs, which contains information of bid requests and DSP responses [22].

Table 6 summarizes the different attributes found in the dataset, which mostly correspond to the OpenRTB specifications. As we will see in Section 4.3, our top-k scoring relies of the prediction of bid values and response times, as described in Section 4.4. In order to enable this prediction, we extract features from the OpenRTB specifications, and train our model using the corresponding attributes in the iPinYou dataset. Furthermore, since the accuracy of the model is highly dependent on the workload, we analyze the iPinYou dataset in detail and derive the statistical distribution of important attributes (using SAS JMP[1]), as shown in Table 6 in Appendix A. This analysis is used by our data generator and described further in Section 6.

[1]https://www.jmp.com/en_ca/home.html

4 PROPOSED PUB/SUB SOLUTION

In this section, we first present our new solution which integrates pub/sub with OpenRTB, in order to eliminate no-bid responses. Then, we investigate how to further reduce the number of irrelevant bid responses by formulating the problem definition of minimum bid responses selection. We show how our pub/sub solution can be extended using top-k filtering to filter out bid responses that are too slow or with low values. Finally, we adapt the top-k solution by replacing the score prediction component with a discrete machine learning model.

4.1 Content-Based Filtering of DSPs

Sending bid requests to DSPs who respond with a no-bid message has two major consequences. First, it generates communication overhead since 2 messages need to be exchanged between the AdX and the DSP: the request and the response. Second, the timeout value t_{max} is potentially affected if the DSP is slow to produce the no-bid response, thus increasing the end-to-end latency for that bid request.

Our analysis of the OpenRTB specifications, as well as the iPinYou dataset, reveals that most no-bid responses are related to the content of the bid request, such as unsupported device or unmatched user (error codes 6 and 8 in Table 3 in Appendix A). Therefore, our solution allows the DSPs to expose any constraint on they may have regarding these two aspects in the form of a content-based subscription, thereby filtering DSPs at the AdX side and eliminating no-bid responses with these error codes. Our proposed RTB processing model is shown in Figure 4. This is accomplished by leveraging the attributes *device* and *user* from the OpenRTB specifications, cf. Table 2 in Appendix A. They are also called *User-Agent* and *UserProfileIDs* in the iPinYou dataset, respectively (cf. Table 6 in Appendix A). Compared to the original model in Figure 2, the following steps have been modified:

Figure 5: Processing Flow with Pub/Sub and Top-K Filtering

Figure 6: Scoring-Based Top-K Overview

(1a) Prior to receiving auctions, DSPs send to the AdX (equipped with a publish/subscribe broker) the following subscription:

$$Subscription : (device, =, val1), (user, =, val2)$$

where *device* is the desired user device (mobile, desktop computer, set top box, etc.), and *user* refers to characteristics of the target audience (keywords, interests, gender, etc.).

(2) Once a user visits a page web, the Publisher delivers to the AdX a publication containing the ad request:

$$Publication : (device, val1), (user, val2)$$

(3) For each inbound ad request, AdX generates a bid request as follows:

$$Bidrequest : (id, site, device, user)$$

The AdX matches the bid request publication against stored subscriptions and forwards it to interested DSPs, which calculates bid prices based on their campaigns. DSPs with unmatched subscriptions will not receive the bid request.

Note that the publish/subscribe system allows each DSP to send multiple subscriptions if necessary. Furthermore, subscriptions can be modified at any time to reflect updated interests from the DSP.

4.2 Optimal Number of Bid Responses

We now focus on the problem of filtering DSPs beyond those that are not interested. Intuitively, the AdX should try to obtain a high winning price for each auction, while waiting just long enough to collect this winning price (which is the second-highest bid in a Vickrey auction). This means that DSPs with either low bid values or slow responses should also be filtered by our proposed solution. Note that this problem is not simply finding the set of DSPs to obtain the maximum winning price or the one with the shortest auction time, since these solutions could have a very long auction time or low winning price, respectively. Instead, we define a general model which can capture both dimensions using a cost model. We formalize our optimization problem of minimum bid responses selection with high second price and low wait time as follows:

Given a set of bid responses $BR = \{(b_1, t_1), (b_2, t_2), ..., (b_n, t_n)\}$ *where* n *is the number of DSPs,* t_i *is response time,* b_i *bid price of* DSP_i *and* BR^c *is a set of chosen bid responses*:

$$\textbf{Minimize } |BR^c|, \textit{ where } BR^c \subseteq BR$$
$$\textbf{Subject to } U(BR^c) \geq U(BR^d), \forall BR^d \subseteq BR$$
$$U(BR_i) = w_1 \times second\_price(BR_i) + w_2 \times (t_{max} - max\_time(BR_i))$$
$$second\_price(BR_i) = max_{j \neq i} b_j, \textit{ where}$$
$$b_i > max_{j \neq i} b_j, \forall (b_i, t_i), (b_j, t_j) \in BR_i$$
$$max\_time(BR_i) = max(t_i), \forall (b_i, t_i) \in BR_i$$

$$w_1 + w_2 = 1$$

In the above formula, $U(BR)$ represents the utility of a subset of bid responses, which is calculated by factoring in the two objective metrics: paid price (second-highest price) and auction time (time of the slowest response received). These two metrics are normalized using weight parameters w_1 and w_2, which can be adjusted depending on the application. The optimal set of chosen DSP is the one that yield the greatest utility, among all possible subsets of DSPs in the system, with the fewest number of DSPs selected.

Given an oracle with perfect knowledge, which can accurately return the minimum bid responses with best bid prices and time taken by each DSP prior to the auction, we can prove that the optimal solution has always size two:

Theorem 1. *The number of chosen DSPs is always 2 (*$|BR^c| = 2$*).*

Proof. (By contradiction) Suppose that the optimal set of chosen bid responses is $BR^c = (b_i, t_i), (b_s, t_s), (b_w, t_w)$. Thus, $|BR^c| = 3$ and $U((b_i, t_i), (b_s, t_s), (b_w, t_w))$ is the maximum among all possible sets of chosen bid responses.

Case 1: Without loss of generality, if $b_i < second\_price(BR^c)$, then $b_i < b_s < b_w$ and $t_i \leq max - time(BR^c) \Rightarrow (b_i, t_i)$ can be removed without affecting utility, since it does not contribute to raising the winning price (second highest price), nor can it increase the response time of remaining DSPs in the set, since each response time is independent.

Case 2: $\forall b, b \geq second\_price(BR^c)$, then $\exists b_s$ and $b_w \geq second\_price \Rightarrow (bi, ti)$ can be also removed, since it means at least 2 of the 3 bids have the same value, so removing one of the three will not affect the value of the second price. Again, removing a bid cannot increase the response time of remaining DSPs.

In both cases, $U((bi, ti), (bs, ts), (bw, tw)) = U((bs, ts), (bw, tw))$, then $|BR^c| = |BR^c - (bi, ti)| = 2 < 3$. □

Therefore, our problem can be represented as a selection of 2 bid responses from n elements, where the utility of the chosen subset is maximal among all subsets of size 2. Thus, it is a combination problem with $\binom{n}{2}$. This combination can be solved in quadratic time $O(n^2)$ [14].

4.3 Top-K Filtering

In the previous section, we demonstrate that the AdX should optimally select 2 DSPs for every bid request. In practice, this is not

Figure 7: Rank-Based Top-K Overview

Figure 8: Binary Selection-Based Top-K Overview

feasible without perfect knowledge of future bid values and response times. Therefore, we propose the use of top-k filtering as a general solution to select a subset of DSPs using a fixed size k, according to a scoring function which models the utility function of the previous section. In ideal circumstances, our top-k solution would yield optimal results if $k = 2$ and the top-k scoring function corresponds perfectly to the utility function $U(BR)$.

Figure 5 shows our proposed solution with pub/sub and top-k filtering. For each arriving bid request b, the AdX compares the subscriptions of n DSPs to obtain $m \leq n$ DSPs matching publication b, and then selecting $k \leq m$ DSPs with the highest score to participate in the auction.

Figure 6 shows how the top-k filtering works in details. The performance of each DSP is predicted in order to calculate their scores. They are then sorted in order to select the highest k scores. The score of a DSP_i for a given request b is calculated with the following formula:

$$score(DSP_i, b) = w_1 \times predicted\_perf(DSP_i, b) - w_2 \times won\_bids(DSP_i) \quad (1)$$

And $predicted\_perf(DSP_i, b)$ is calculated as follows:

$$predicted\_perf(DSP_i, b) = \frac{predicted\_price(DSP_i, b)}{predicted\_time(DSP_i, b)} \quad (2)$$

The above formula is similar to the theoretical formulation as it jointly considers both metrics of price and time. To introduce some fairness in the system, the score substracts *wonBids*, which indicates the number of bids already won by the DSP_i. This reduces the likelihood of a DSP to always be selected and denying the opportunity for others to bid. Finally, w_1 and w_2 are adjustable parameters used to tune the weight of each component of the score.

4.4 Score Prediction Model

To predict the performance of DSPs for any given request, we need to estimate the bidding prices and response times using its historical training data (e.g., iPinYou dataset), which consists of past bid requests (with detailed information about their features) and the corresponding responses from DSPs, including their bid prices and response times. We propose the use of regression analysis to predict price and response time as two continuous variables in $\mathbb{R}_{\geq 0}$.

First, we select features of the iPinYou dataset to use for regression, using the Filter method [8]. Table 7 in Appendix A shows the correlation matrix of features found in the iPinYou dataset.

Regression Technique	Metric	MAE	R^2
Linear regression	Bid price	-3.50	0.88
	Response time	-8.60	0.29
Regression tree	Bid price	-4.37	0.85
	Response time	-6.84	-38.87
KNN regression	Bid price	-35.61	0.29
	Response time	-6.54	-13.57

Table 1: Results of Regression Metrics

Then, we remove every feature that is highly correlated with another feature (correlation coefficient $|c| > 0,5$, shown in red) other than bidding price and response time. We obtain the following remaining features: Timestamp, AdExchange, AdSlotHeight, CityID, AdSlotVisibility, AdSlotFloorPrice and UserProfileIDs.

To predict bid price and response time, we compare three popular regression methods: Linear regression, Regression tree and K-Nearest Neighbors (for regression). For each DSP, we apply these algorithms the dataset and compare the results of each algorithms using the following metrics [12]:

Mean Absolute Error (MAE): The average of the differences between the predictions and the actual values. 0 is a perfect fit.

R^2: An indication of goodness of fit of a set of forecasts compared to actual values, ranging from 0 to 1.

According to the results in Table 1, we note that linear regression performs the best for the bid price with $MAE \simeq -3.50$ and $R^2 \simeq 0.88$. We also find that KNN regression works the best for the response time with $MAE \simeq -6.54$ and $R^2 \simeq -13.57$. Therefore, we will implement these two techniques in our evaluation.

5 DISCRETE PREDICTION MODELS

The proposed top-k solution requires the prediction of two continuous variables: bid price and response time. As we will see in the evaluation (Section 6), the regression analysis is slow to calculate predictions, which negatively impacts the end-to-end latency of running each auction. In order to speed up the prediction, we propose the use of discrete models.

The main insight is that the top-k filtering purpose is solely to determine which DSPs to contact in order to run a profitable and short auction: it does not need to accurately predict the winning bid or total time taken for the auction. Therefore, the current scoring mechanism is calculating more than needed, which degrades

Figure 9: Baseline Comparison for the Number of Bid Responses

performance. We propose two prediction models using discrete variables: rank-based and binary selection-based. In order to use these models, we must adapt the original top-k filtering process, which will be described in each subsection.

5.1 Rank-Based Top-K

This method is based on predicting the rank of each interested DSP by the current bid request (7) (cf. Figure 7). First, we use the same historical data as before, except we calculate the score of each DSP based on their bid value and response time, and use the scores to establish the ranking of the DSPs for each bid request. Then, we train a linear regression algorithm to predict the rank of each DSP given the features of the bid request. Finally, the AdX selects k DSPs with the highest rank.

5.2 Binary Selection-Based Top-K

While the previous solution uses a discrete variable, it still is performing more work than necessary since it tries to accurately predict the rank of each DSP. Our next approach is to classify each DSP in two categories: "yes" or "no", answering solely the question whether a DSP should be contacted or not for a given RTB auction. We accomplish this using classification analysis (cf. Figure 8). We tested five types of classifiers: FeedForward, Bayesian, NEAT, PNN and RBF. We opted to use FeedForward for its superior execution time compared to others. To train this model, we use the same dataset, but convert the score into a "yes" or "no" value based on whether this DSP belongs to the top-k for an auction or not. Note that this method does not guarantee top-k results with exactly k DSPs. Furthermore, the training is done for a specific value of k chosen in advance.

6 EVALUATION

In this section, we experimentally evaluate our solutions: Pub/Sub, and Top-k against a baseline implementation of OpenRTB. Our experiments contains a performance comparison of the three approaches with respect to auction time and paid price. Then, we conduct a sensitivity analysis of the major parameters impacting the solutions.

6.1 Experimental Setup

Implementation: Our work is based on the OpenRTB 2.0 reference implementation[2]. We implemented in J2EE our own AdX according to the OpenRTB API specifications (version 2.5) [6]. We use RabbitMQ using the Header Exchanges as the pub/sub broker [7]. Finally, we add the top-k engine to the AdX and integrate it with RabbitMQ.

Bidding strategies: We employ 100 DSPs distributed across three different types of bidding strategies:

- **Constant**: The reference implementation provides a simple bidder type of DSP, which adds a constant value over the floor price for each request.
- **Random**: We implemented our own random bidding strategy which chooses a value between the floor price and a parametric upper bound.
- **Below max eCPC**: We implemented the strategy discussed in [21] where the bid price is calculated by multiplying the max eCPC (effective cost per click) and its predicted CTR (click-through rate).

Default parameters: We set the selectivity to 70% of interested DSPs per bid request. Our default top-k method is the scoring-based one (cf. Section 4.3), with $k = 10$.

Workload: By default, we employ the iPinYou dataset, in particular the logs containing auction data (bid requests and responses). For the sensitivity analysis, we developed a dataset generator using iPinYou datset distributions with adjustable parameters. We generate 100 bid requests at a time.

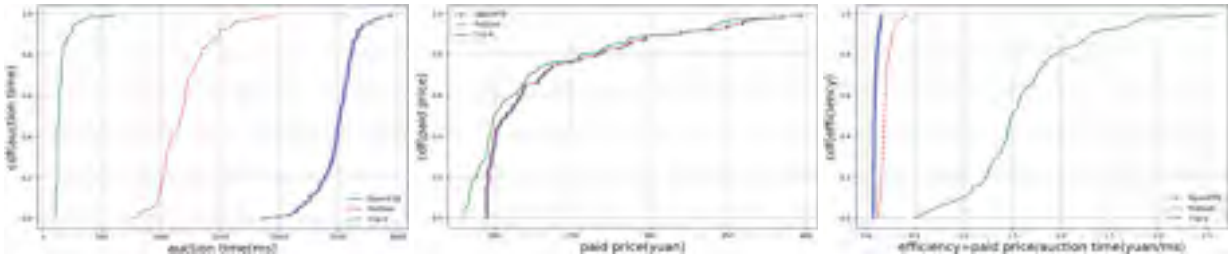
Environment: We conduct our evaluation using an Ubuntu virtual machine version 18.04.3 LTS, with 4096 MB of RAM and 2 GHz CPU.

6.2 Performance Comparison

Using four different metrics, we compare our three approaches: OpenRTB (baseline), Pub/Sub only, and Pub/Sub with Top-k.

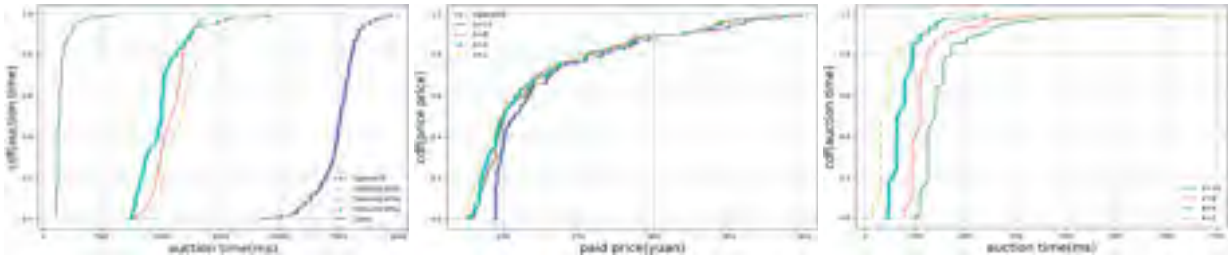
Number of messages: Figure 9 shows the number of bid responses sent by DSPs and the rate of no-bids responses. Compared to the baseline, the Pub/Sub and Top-k reduce the total number of messages by 47% and 90%, respectively. In particular, 4696 no-bid responses are received by the AdX, which are completely eliminated by both of our solutions. The top-k solution can further filter out an

[2] https://github.com/openrtb/openrtb2x



(a) Auction Time (b) Paid Price (c) Efficiency

Figure 10: Baseline Comparison between OpenRTB, Pub/Sub, and Scoring-Based Top-K



(a) Selectivity vs. Auction Time (b) k vs. Paid Price (c) k vs. Auction Time

Figure 11: Sensitivity Analysis of the Scoring-Based Top-K Solution

additional 4304 messages compared to only pub/sub, which are low-scoring bid responses. Furthermore, the top-k filtering increases stability, since exactly 10 responses are received for each of the 100 bid requests.

Auction time: Figure 10a shows the cumulative distribution function (CDF) for the end-to-end auction time of bid requests processed by the three approaches. The auction time is measured from the moment the AdX receives the bid request from the publisher to the moment it contacts the winning DSP. In the OpenRTB baseline implementation, auctions takes between $1800ms$ to $3000ms$, compared to $800 - 2,000ms$ for pub/sub, and $120 - 600ms$ for top-k. The median for each solution is $2510ms$, $1171ms$ and $140ms$ for baseline, pub/sub, and top-k respectively. Our pub/sub and top-k solutions are therefore able to reduce the auction time by 50% and 94%. This result demonstrates that a reduction in the number of DSPs contacted by the AdX decreases the chance of encountering a slow DSP (straggler), thereby reducing the overall time.

Paid price: Figure 10b shows the CDF for the paid price using the same bid requests for all three approaches. Here, the baseline OpenRTB is optimal since it contacts every DSP every time, guaranteeing the maximum price of each auction. The pub/sub solution also performs optimally, since it only eliminates no-bid responses which no chance of winning. For the top-k solution, a median loss of 3.75% is incurred for two reasons. First, a DSP with high bid price but slow response time might not be selected as it will score poorly for top-k. Second, the prediction model used might incorrectly underestimate the score of a DSP, causing it not to be selected when it would have impacted the paid price (i.e., submitted a bid higher than the second highest price).

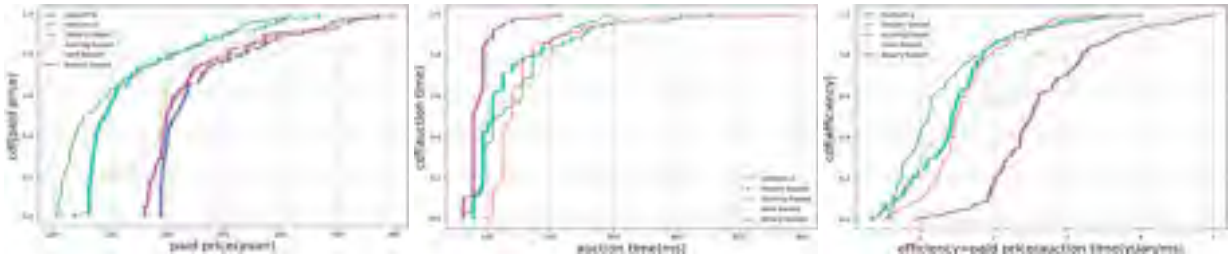
Efficiency: Figure 10c evaluates the efficiency of the three solutions. This metric combines the previous two metrics by dividing the paid price with the auction time. Since the pub/sub solution yields the same paid prices as OpenRTB with shorter auction times, its efficiency is superior with a median of $\simeq 0,19$ yuan/ms against only $\simeq 0,08$ yuan/ms, a 2.375 times increase. However, top-k is even better with a median efficiency of $\simeq 1.48$ yuan/ms, which is 18.5 times better than the baseline and 7.79 times better than the pub/sub only solution. Although top-k sacrifices a small loss in paid prices, its execution time is also substantially shorter. The trade-off is therefore in favor of the top-k solution.

Summary: Compared to the baseline, the pub/sub and top-k approaches reduce the number of messages by 47% and 90%, and the auction time by 50% and 94%, respectively. The pub/sub only approaches does not compromise on the paid price, while the top-k approach suffers a loss of 3.61% in paid prices. However, this loss is offset by a huge speedup in auction time, as evidenced by the superior efficiency of the top-k approach. The top-k approach is therefore desirable if the application receives a near-infinite stream of requests that must be treated efficiently. On the other hand, if the volume of bid requests is limited, the pub/sub only approach is desirable in order to extract the maximum price per auction.

6.3 Sensitivity Analysis

We conduct a sensitivity analysis for our solutions in order to study the impact of three parameters: selectivity of bid requests, value of k, and choice of prediction model.

Selectivity: Figure 11a shows the impact of varying the selectivity of DSPs. This parameter controls how many DSPs are interested in each bid request. We test 3 scenarios: 30%, 50% and 70%. The



(a) Top-K Model vs. Paid Price (b) Top-K Model vs. Auction Time (c) Top-K Model vs. Efficiency

Figure 12: Comparison of Top-K Solutions

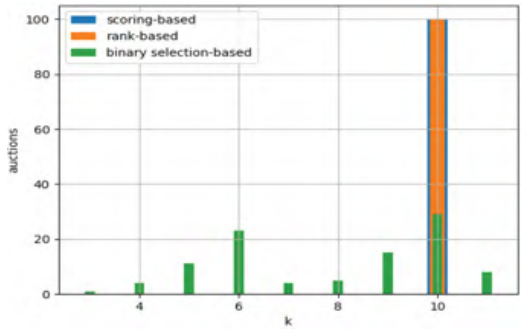


Figure 13: Number of DSPs Selected for Top-K Solutions

baseline solution is unaffected by this parameter, since it contacts every DSP without considering their interests. The top-k solution is also insensitive to selectivity: if k is sufficiently low, the change in sensitivity does not impact the number of DSPs selected, which is always k per bid request. Finally, the pub/sub only approach is affected the selectivity: the median are 999ms, 1083ms, and 1171ms for 30%, 50%, and 70%. As the selectivity increases, the filtering performed by the pub/sub broker decreases, which reduces the effectiveness of the solution.

Parameter k: We test 4 different values of k: 2, 4, 8, and 10. Figure 11b shows the paid price CDF for each scenario, compared to the optimal price represented by the OpenRTB baseline. As expected, the paid price generally decreases when k decreases, since the margin of error becomes narrower for the prediction model to correctly identify the ideal DSPs for the auction. However, the difference is not substantial, with a median loss of 4.1% for $k = 2$ compared to 3.61% for $k = 10$. This indicates that the regression model used is highly accurate.

For the auction time, Figure 11c shows that a noticeable improvement can be achieved by reducing the value of k. The median value for $k = 2$ is $32ms$ compared to $140ms$ to $k = 10$.

Prediction models: We compare our three prediction models: scoring-based, rank-based, and binary selection-based, with two baselines: history-based (average of past auctions) and random k (choose k DSPs at random).

Figure 12a shows the CDF for paid prices. Both baselines (history and random) are clearly inferior to our proposed solution, with intervals of 120−340 yuan and 100−310 yuan. One notable exception is the rank-based one, which obtained poor prices for 20% of the auctions and thus has a wide interval of 100-390 yuan. On the other hand, the paid prices for scoring-based and binary selection-based are in the range of $180 - 390$ yuan, which is near optimal to the OpenRTBbase line.

Figure 12b shows the CDF for auction time. Random-k and history-based have an interval range of $40 - 300ms$ and $70 - 450ms$. The scoring-based model is the heaviest to compute, with an interval of $100 - 600ms$. The rank-based model has an interval of $40 - 330ms$ and the binary selection-based top-k outperforms all others with an interval of $40 - 220ms$.

We compare also the efficiency (paid price/auction time) in Figure 12c which clearly demonstrates that the binary selection-based is the most efficient model with a range of $0.9 - 5$ yuan/ms compared to other models that are almost in the same interval $0.5 - 4$ yuan/ms.

Since the binary selection-based model does not guaranteed a fixed k per bid request, we also measure the variation in the size of the sets returned in Figure 13. The results show that the set vary between 3 to 11 DSPs selected. For rank-based and scoring-based, the results confirm that the number of DSPs selected is always k (10).

Summary: When compared to the baseline and the pub/sub solution, the top-k solution stands out as being the most reliable, since it is not sensitive to the selectivity of the bid requests. For the scoring-based solution, k can be set to a surprisingly low number (as low as 2), with little loss of price and substantial speedup in auction time. When comparing prediction models for top-k solutions, each of three proposed approaches have advantages and disadvantages, as highlighted in Table 2. For maximum efficiency, the binary-based selection model stands out as it is noticeably faster with minimal price loss, with the drawback of returning uneven-sized results and being inflexible. The scoring-based model has the worst efficiency, but can obtain better prices than other solutions. The rank-based solution is a compromise between the two, as it has average efficiency and retains flexibility. However, it suffers from uneven performance for a minority of requests (20%).

7 CONCLUSIONS

The current standard for real-time bidding (RTB) broadcasts bid requests to all DSP bidders, which generates massive communication and computation overhead. We propose the use of publish/subscribe

Top-K Models	Advantages	Disadvantages
Scoring-based	Near-optimal prices; flexible choice of k	Poor execution time; loss of efficiency
Rank-based	Flexible choice of k	Uneven performance
Binary selection-based	Best execution time and efficiency	Variable size of results; the value of k is fixed during training

Table 2: Summary of Top-k Filtering Solutions

to express the interests of DSP bidders as content-based subscriptions, in order to eliminate no-bid responses through auctioning with selected DSPs only.

Furthermore, we explore how to further reduce the number of DSPs contacted by avoiding bids with low prices or slow response times. We formulate the problem of optimal number of bid responses, and demonstrate how top-k filtering can be used to address this problem in a online setting. Top-k filtering relies on a prediction model in order to guess which DSPs should participate in which auction. Our scoring-based model uses regression analysis with continuous variables, while our rank-based and binary selection-based models both employ discrete analysis.

Our evaluation confirms the effectiveness of our approach in reducing the number of messages required and the auction time, while maintaining optimal or near-optimal winning prices. Our most effective solution is the OpenRTB implementation using Pub/Sub with a binary selection-based top-k filtering mechanism.

For our future work, we wish to test the applicability of our approach across a wider range of datasets beyond iPinYou, and to implement more sophisticated bidding strategies which could challenge the accuracy of our prediction models. We will also investigate making our solution self-adaptive, tuning its parameters (e.g., value of k) by monitoring the current conditions.

REFERENCES

[1] Anonymized.
[2] Anonymized.
[3] Anonymized.
[4] Chen, L., Cong, G., Cao, X., and Tan, K. L. Temporal Spatial-Keyword Top-k publish/subscribe. *Proceedings - International Conference on Data Engineering 2015-May* (2015), 255–266.
[5] Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. The many faces of publish/subscribe. *ACM Computing Surveys 35*, 2 (2003), 114–131.
[6] IAB Technology Laboratory. *OpenRTB API Specification Version 2.5*, 2016.
[7] Ionescu, V. M. The analysis of the performance of RabbitMQ and ActiveMQ. *2015 14th RoEduNet International Conference - Networking in Education and Research, RoEduNet NER 2015 - Proceedings* (2015), 132–137.
[8] KAUSHIK, S. Introduction to feature selection methods with an example.
[9] Kumar, J. Timeout Analysis, Troubleshooting and Notification in Real Time Bidding Advertising System with Implementation. *Computer Science and Engineering 7*, 3 (2017), 67–78.
[10] Lee, K. C., Jalali, A., and Dasdan, A. Real time bid optimization with smooth budget delivery in online advertising. *Proceedings of the 7th International Workshop on Data Mining for Online Advertising, ADKDD 2013 - Held in Conjunction with SIGKDD 2013* (2013).
[11] Liao, H., Peng, L., Liu, Z., and Shen, X. iPinYou Global RTB Bidding Algorithm Competition Dataset. 1–6.
[12] Moayedi, H., Bui, D. T., Dounis, A., and Lyu, Z. applied sciences Predicting Heating Load in Energy-E ffi cient Buildings Through Machine Learning Techniques.
[13] Mullarkey, M. T., and Hevner, A. R. New Horizons in Design Science: Broadening the Research Agenda. *Desrist 2015 LNCS 9073* (2015), 121–134.
[14] Oppen, D. C. Complexity, convexity and combinations of theories. *Theoretical Computer Science 12*, 3 (1980), 291 – 302.
[15] Qin, R., Yuan, Y., and Wang, F. Y. Optimizing the revenue for ad exchanges in header bidding advertising markets. *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017 2017-January* (2017), 432–437.
[16] Sayedi, A. Real-time bidding in online display advertising. *Marketing Science 37*, 4 (2018), 553–568.
[17] Shraer, A., Gurevich, M., Fontoura, M., and Josifovski, V. Top-k publish-subscribe for social annotation of news. *Proceedings of the VLDB Endowment 6*, 6 (2014), 385–396.
[18] Wang, J., Zhang, W., and Yuan, S. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting.
[19] Wu, W. C. H., Yeh, M. Y., and Chen, M. S. Predicting winning price in real time bidding with censored data. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2015-August* (2015), 1305–1314.
[20] Yuan, Y., Wang, F., Li, J., and Qin, R. A survey on real time bidding advertising. *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2014* (2014), 418–423.
[21] Zhang, C. R., and Zhang, E. Optimized bidding algorithm of real time bidding in online ads auction. *International Conference on Management Science and Engineering - Annual Conference Proceedings* (2014), 33–42.
[22] Zhang, W., Yuan, S., and Wang, J. Real-Time Bidding Benchmarking with iPinYou Dataset.

A APPENDIX

Code	No-Bid Reason
0	Unknown error
1	Technical error
2	Invalid error
3	Known web spider
4	Suspected Non-human traffic
5	Cloud, data center or proxy IP
6	Unsupported device
7	Blocked publisher or site
8	Unmatched user
9	Daily reader cap met
10	Daily domain cap met

Table 3: No-Bid Reasons in OpenRTB

Attribute	Type	Description
id	String; required	Unique ID of the bid request
imp	Object array; required	Array of impressions offered
site	Object; recommended for websites	Details about publisher's website
app	Object; recommended for apps	Details about publisher's app
device	Object; recommended	Details about user's device
user	Object; recommended	Details about the advertising audience
test	Integer; default 0	Indicator of test mode in which auctions are not billable (0=live mode, 1=test mode)
at	Integer; default 2	Auction type, where 1=first price, 2=second price
tmax	Integer	timeout in ms
wseat	String array	White list of buyer seats allowed to bid on this impression

Table 4: Bid Request Attributes in OpenRTB

Attribute	Type	Description
id	String; required	Unique ID of the bid response
seatbid	Object array	Array of seatbid objects
site	Object; recommended for websites	Details about publisher's website
bidid	String	Bidder generated response ID
cur	String; default "USD"	Bid currency using ISO4217 alpha codes
customdata	String	Optional feature to allow a bidder to set data in exchanger's cookie
nbr	Integer	Reason for not bidding
ext	Object	Placeholder for bidder

Table 5: Bid Response Attributes in OpenRTB

Feature	Description	Distribution
BidID	Identity of the ad impression	Pseudorandom String [32] of (a..z) and (0..9)
Timestamp	Arrival time of the bid request at the DSP	Normal distribution
Log Type	Type of the log: 1 (impression), 2 (click), or 3 (conversion)	Constant distribution
User Agent	Device through which the user visits the website	Random type of device (e.g, Windows NT 6.1)
(RegionID,CityID)	Location of the user	Binomial Beta distribution with $\mu = 66$ and $\sigma = 77$
Ad Exchange	Source of the ad impression	Binomial Beta distribution with $\mu = 2.2388$ and $\sigma = 0.7821$
(AdSlotWidth, AdSlotHeight)	Size of the ad slot	Negative binomial distribution with $\mu = 3.527$ and $\sigma = 2.329$
AdSlotVisibility	Visibility of the ad (with regards to page scrolling): (1) Above the page fold, (2) Below the page fold, (0) Unknown	Binomial Beta distribution with $\mu = 0.867$ and $\sigma = 0.939$
AdSlotFormat	Delivery format: (1) Fixed ad slot, (2) Popup window, (0) Unknown	Constant distribution
AdSlotFloorPrice	Minimum price of the winning bid	Zero-inflated negative binomial distribution with $\mu = 28.977$ and $\sigma = 43.709$
AdvertiserID	Advertiser identity	Constant distribution
UserProfileIDs	Identities of DAAT [11] (Digital Audience Advertising Taxonomy) category names	Negative binomial distribution with $\mu = 10804.419$ and $\sigma = 1622.4653$
User-Agent	User device and browser	Not available (textual data)

Table 6: Important Characteristics of the iPinYou Dataset

	(f1)	(f2)	(f3)	(f4)	(f5)	(f6)	(f7)	(f8)	(f9)	(f10)	(f11)
Timestamp (f1)	1.00	-0.28	0.03	-0.04	0.18	0.18	-0.08	-0.34	-0.30	0.09	-0.03
AdExchange (f2)	-0.29	1.00	-0.13	-0.12	0.08	0.07	0.03	0.14	0.15	0.10	0.04
AdSlotHeight (f3)	0.29	-0.13	1.00	0.60	-0.19	-0.19	0.16	-0.01	0.03	-0.30	-0.01
AdSlotWidth (f4)	-0.04	-0.12	0.60	1.00	-0.18	-0.17	0.16	0.005	-0.01	-0.16	0.01
RegionID (f5)	0.18	0.08	-0.19	0.18	1.00	0.99	0.005	-0.16	-0.14	0.19	0.04
CityID (f6)	0.18	0.07	-0.19	-0.17	0.99	1.00	0.01	-0.17	-0.15	0.18	0.04
AdSlotVisibility (f7)	-0.08	0.03	0.16	0.16	0.06	0.01	1.00	-0.09	-0.09	-0.11	0.02
AdSlotFloorPrice (f8)	-0.34	0.14	-0.01	0.005	-0.16	-0.17	-0.09	1.00	0.97	-0.02	-0.01
BiddingPrice (f9)	-0.30	0.15	-0.03	-0.01	-0.14	-0.15	0.09	0.97	1.00	0.01	-0.01
ResponseTime (f10)	0.09	0.11	-0.29	-0.16	0.19	0.18	-0.11	-0.02	0.01	1.00	0.01
UserProfileIDs (f11)	-0.30	0.04	-0.01	0.01	0.04	0.04	0.02	-0.01	-0.01	0.01	1.00

Table 7: Correlation Matrix for iPinYou Dataset

ANNEXE II

CODAGE DE FONCTIONNALITÉS DU PROJET DE RECHERCHE

Code principal du DSP SIMPLE ordinaire

```
public BidResponse process(BidRequest request) throws
   AvroRemoteException{
     BidResponse response = null;
     if (validateRequest(request)){
       RTBRequestWrapper wReq = (RTBRequestWrapper)request;
       response = new BidResponse();
       response.id = wReq.getId();
       response.bidid = "simple-bid-tracker";
       Map<String, String> seats =
          wReq.getUnblockedSeats(wReq.getSSPName());
       for (Impression i : wReq.getRequest().getImp()){
         for (Map.Entry<String, String> s : seats.entrySet()){
           RTBAdvertiser a = wReq.getAdvertiser(s.getValue());
           String device=wReq.device.toString();
           String[] user_profiles=wReq.getUser().getKeywords().
             toString().split(",");
           SeatBid seat_bid = new SeatBid();
           seat_bid.seat = s.getKey();
           seat_bid.bid = new ArrayList<Bid>();
           Bid b = new Bid();
           b.id = "SimpleBid#" + lastBidNum++;
           b.impid = i.getId();
           b.price= (float) 0.0;
           for(String k: a.getUser_profiles()){
             for(CharSequence user_profile:user_profiles){
               if(k.equals(user_profile.toString())){
```

```
                if(a.getDevice().equals(device)) {
                  b.price = i.getBidfloor()+n + 100;
                  a.minBudget(b.price);
                }
              }
            }
          }
          b.nurl = a.getNurl();
          b.adid = adId; // serves up the same ad to all
             impressions
          seat_bid.bid.add(b);
          List<SeatBid> list = new ArrayList<SeatBid>();
          list.add(seat_bid);
          response.setSeatbid(list);
        }
      }
    }
    return response;
  }
```

Code principal du DSP SIMPLE après l'utilisation de Publish/Subscribe

```
public BidResponse process(BidRequest request) throws
   AvroRemoteException{
    BidResponse response = null;
    if (validateRequest(request)) {
      RTBRequestWrapper wReq = (RTBRequestWrapper)request;
      response = new BidResponse();
      response.id = wReq.getId();
      response.bidid = "simple-bid-tracker";
```

```
        Map<String, String> seats =
            wReq.getUnblockedSeats(wReq.getSSPName());
        for (Impression i : wReq.getRequest().getImp()){
          for (Map.Entry<String, String> s : seats.entrySet()){
            RTBAdvertiser a = wReq.getAdvertiser(s.getValue());
            SeatBid seat_bid = new SeatBid();
            seat_bid.seat = s.getKey();
            seat_bid.bid = new ArrayList<Bid>();
            Bid b = new Bid();
            b.id = "SimpleBid#" + lastBidNum++;
            b.impid = i.getId();
            b.price = i.getBidfloor()+n + 100;
            a.minBudget(b.price);
            b.nurl = a.getNurl();
            b.adid = adId; // serves up the same ad to all
               impressions
            seat_bid.bid.add(b);
            List<SeatBid> list = new ArrayList<SeatBid>();
            list.add(seat_bid);
            response.setSeatbid(list);
          }
        }
      }
      return response;
    }
```

Code principal du DSP RANDOM ordinaire

```
public BidResponse process(BidRequest request) throws
   AvroRemoteException{
     BidResponse response = null;
```

```
if (validateRequest(request)){
  RTBRequestWrapper wReq = (RTBRequestWrapper)request;
  response = new BidResponse();
  response.id = wReq.getId();
  response.bidid = "simple-bid-tracker";
  Map<String, String> seats =
     wReq.getUnblockedSeats(wReq.getSSPName());
  for (Impression i : wReq.getRequest().getImp()){
    for (Map.Entry<String, String> s : seats.entrySet()){
      RTBAdvertiser a = wReq.getAdvertiser(s.getValue());
      String device=wReq.device.toString();
      String[] user_profiles=wReq.getUser().getKeywords().
         toString().split(",");
      SeatBid seat_bid = new SeatBid();
      seat_bid.seat = s.getKey();
      seat_bid.bid = new ArrayList<Bid>();
      Bid b = new Bid();
      b.id = "SimpleBid#" + lastBidNum++;
      b.impid = i.getId();
      b.price= (float) 0.0;
      Random ran=new Random();
      ran.setSeed(seed);
      price=ran.nextFloat()*100;
      for(String k: a.getUser_profiles()){
        for(CharSequence user_profile:user_profiles){
          if(k.equals(user_profile.toString())){
            if(a.getDevice().equals(device)) {
              b.price = i.getBidfloor()+n +(float) price;
              a.minBudget(b.price);
            }
          }
```

```
            }
          }

          b.nurl = a.getNurl();
          b.adid = adId; // serves up the same ad to all
             impressions
          seat_bid.bid.add(b);
          List<SeatBid> list = new ArrayList<SeatBid>();
          list.add(seat_bid);
          response.setSeatbid(list);
        }
      }
    }
    return response;
  }
```

Code principal du DSP RANDOM après l'utilisation de Publish/Subscribe

```
public BidResponse process(BidRequest request) throws
   AvroRemoteException{
    BidResponse response = null;
    if (validateRequest(request))
    {  RTBRequestWrapper wReq = (RTBRequestWrapper)request;
    response = new BidResponse();
    response.id = wReq.getId();
    response.bidid = "random-bid-tracker";
    Map<String, String> seats =
       wReq.getUnblockedSeats(wReq.getSSPName());
    for (Impression i : wReq.getRequest().getImp()){
      for (Map.Entry<String, String> s : seats.entrySet()){
        RTBAdvertiser a = wReq.getAdvertiser(s.getValue());
```

```
          SeatBid seat_bid = new SeatBid();
          seat_bid.seat = s.getKey();
          seat_bid.bid = new ArrayList<Bid>();
          Bid b = new Bid();
          b.id = "RandomBid#" + lastBidNum++;
          b.impid = i.getId();
          Random ran=new Random();
          ran.setSeed(seed);
          price=ran.nextFloat()*100;
          b.price = i.getBidfloor()+n +(float) price;
          b.nurl = a.getNurl();
          b.adid = adId; // serves up the same ad to all impressions
          seat_bid.bid.add(b);
          List<SeatBid> list = new ArrayList<SeatBid>();
          list.add(seat_bid);
          response.setSeatbid(list);
        }

      }
      }
      return response;
    }
```

Code principal du DSP Below_eCPC

```
public BidResponse process(BidRequest request) throws
    AvroRemoteException{
      BidResponse response = null;
      if (validateRequest(request)) {
        RTBRequestWrapper wReq = (RTBRequestWrapper)request;
        response = new BidResponse();
```

```
response.id = wReq.getId();
response.bidid = "belowCPC-bid-tracker";
Map<String, String> seats =
   wReq.getUnblockedSeats(wReq.getSSPName());
for (Impression i : wReq.getRequest().getImp()){
  for (Map.Entry<String, String> s : seats.entrySet()){
    RTBAdvertiser a = wReq.getAdvertiser(s.getValue());
    String[] user_profiles=wReq.getUser().getKeywords().
       toString().split(",");
    SeatBid seat_bid = new SeatBid();
    seat_bid.seat = s.getKey();
    seat_bid.bid = new ArrayList<Bid>();
    Bid b = new Bid();
    b.id = "BelowMaxCpcBid#" + lastBidNum++;
    b.impid = i.getId();
    b.price= (float) 0.0;
    for(String k: a.getUser_profiles()){
      for(CharSequence user_profile:user_profiles){
        if(k.equals(user_profile.toString())){
          b.price = (float) (maxCPC*ctr);
          a.minBudget(b.price);
        }
      }
    }

    b.nurl = a.getNurl();
    b.adid = adId; // serves up the same ad to all
       impressions
    seat_bid.bid.add(b);
    List<SeatBid> list = new ArrayList<SeatBid>();
    list.add(seat_bid);
```

```
            response.setSeatbid(list);
          }
        }
      }
      return response;
    }
```

Code principal du DSP Below_eCPC après l'utilisation de Publish/Subscribe

```
public BidResponse process(BidRequest request) throws
    AvroRemoteException{
      BidResponse response = null;
      if (validateRequest(request)){
        RTBRequestWrapper wReq = (RTBRequestWrapper)request;
        response = new BidResponse();
        response.id = wReq.getId();
        response.bidid = "belowCPC-bid-tracker";
        Map<String, String> seats =
           wReq.getUnblockedSeats(wReq.getSSPName());
        for (Impression i : wReq.getRequest().getImp()){
          for (Map.Entry<String, String> s : seats.entrySet()){
            RTBAdvertiser a = wReq.getAdvertiser(s.getValue());
            SeatBid seat_bid = new SeatBid();
            seat_bid.seat = s.getKey();
            seat_bid.bid = new ArrayList<Bid>();
            Bid b = new Bid();
            b.id = "BelowCPCBid#" + lastBidNum++;
            b.impid = i.getId();
            b.price =(float) (maxCPC*ctr);
            a.minBudget(b.price);
            b.nurl = a.getNurl();
```

```
            b.adid = adId; // serves up the same ad to all
               impressions
            seat_bid.bid.add(b);
            List<SeatBid> list = new ArrayList<SeatBid>();
            list.add(seat_bid);
            response.setSeatbid(list);
          }
        }
      }
      return response;
    }
```

Code du générateur de requêtes d'enchères

```
package openrtb.rtb_app.generator;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

import org.apache.commons.math3.distribution.BetaDistribution;
import org.apache.commons.math3.distribution.BinomialDistribution;
import org.apache.mahout.common.RandomUtils;
import org.apache.mahout.math.jet.random.NegativeBinomial;

import umontreal.iro.lecuyer.randvar.HalfNormalGen;
import umontreal.iro.lecuyer.rng.RandomStreamBase;

public class RequestGenerator {
```

```
public String generateBidID(){
  String chars = "abcdefghijklmnopqrstuvwxyz1234567890";
  String pass = "";
  for(int x=0;x<32;x++){
    int i = (int)Math.floor(Math.random() * 36);
    pass += chars.charAt(i);
  }
  return(pass);
}
public String generateTimestamp() {
  double var=Math.pow(48964764, 2);
  double moy=108866977;
  int n=174740000;
  double alpha=(((n*moy)- Math.pow(moy, 2)-
     var)/((var*n/moy)-n+moy));
  double beta= alpha * ((n / moy) - 1);
  double b1;
  int b2;
  BetaDistribution bet = new BetaDistribution(alpha,beta);
  b1 = bet.sample();
  BinomialDistribution binom = new BinomialDistribution(n,b1);
  b2 = binom.sample();
  String result=String.valueOf(b2);
  return(result);

}

public String generateRegionCity() throws IOException {
  double moy=66.000149;
  double ecart=77.067621;
  int n=332;
```

```
    double var = Math.pow(ecart,2);
    double alpha=(((n*moy)- Math.pow(moy, 2)-
       var)/((var*n/moy)-n+moy));
    double beta= alpha * ((n / moy) - 1);
    double b1;
    int b2;
    BetaDistribution bet = new BetaDistribution(alpha,beta);
    b1 = bet.sample();
    BinomialDistribution binom = new BinomialDistribution(n,b1);
    b2 = binom.sample();
    UnconvertRegionCity u=new UnconvertRegionCity();
    return(u.Unsort_lookup(u.Sort_unsort(String.valueOf(b2))));
  }
  public int generateAdExchange() {
    double var=Math.pow(0.7821843, 2);
    double moy=2.2388489;
    int n=3;
    double alpha=(((n*moy)- Math.pow(moy, 2)-
       var)/((var*n/moy)-n+moy));
    double beta= alpha * ((n / moy) - 1);
    double b1;
    int b2;
    BetaDistribution bet = new BetaDistribution(alpha,beta);
    b1 = bet.sample();
    BinomialDistribution binom = new BinomialDistribution(n,b1);
    b2 = binom.sample();
    return(b2);
  }
  public String generateAdSlotSize() throws IOException {
    double moy=3.527226;
    double var = Math.pow(2.3290767,2);
```

```
    double p = (var - moy) / var;
    int r = (int) (Math.pow(moy,2) / (var - moy));
    int b=13;
    while(b>12) {
      NegativeBinomial nbinom = new NegativeBinomial(r,p, new
         Random());
      b=nbinom.nextInt();
    }
    UnconvertAdSlotSize u=new UnconvertAdSlotSize();
    return(u.Unsort_lookup(u.Sort_unsort(String.valueOf(b))));
  }
  public int generateAdSlotVisibility() {
    double var=Math.pow(0.9398591, 2);
    double moy=0.8679063;
    int n=2;
    double alpha=(((n*moy)- Math.pow(moy, 2)-
       var)/((var*n/moy)-n+moy));
    double beta= alpha * ((n / moy) - 1);
    double b1;
    int b2;
    BetaDistribution bet = new BetaDistribution(alpha,beta);
    b1 = bet.sample();
    BinomialDistribution binom = new BinomialDistribution(n,b1);
    b2 = binom.sample();
    return(b2);
  }
  public double generateAdSlotFloorPrice() {
    double moy=32.998936;
    double sigma=43.809043;
    double var=Math.pow(sigma, 2);
    int n=240;
```

```
    double alpha=(((n*moy)- Math.pow(moy, 2)-
       var)/((var*n/moy)-n+moy));
    double beta= alpha * ((n / moy) - 1);
    double b1;
    int b2;
    BetaDistribution bet = new BetaDistribution(alpha,beta);
    b1 = bet.sample();
    BinomialDistribution binom = new BinomialDistribution(n,b1);
    b2 = binom.sample();
    return(b2);
  }

  public String generateUserProfileID() {
    String userProfileAffich="";
    double mu=10804.419;
    double var = Math.pow(1622.4653,2);
    double p = (var - mu) / var;
    int r = (int) (Math.pow(mu,2) / (var - mu));
    for(int i=0;i<randomInt;i++){
      NegativeBinomial nbinom = new
         NegativeBinomial(r,p,RandomUtils.getRandom());
      String s=nbinom.nextInt(r,p);
      userProfileIds[i]=s;
      userProfileAffich=userProfileAffich+s+",";
      }
    }
    userProfileAffich=userProfileAffich+"ff";
    return (userProfileAffich.replaceAll(",ff",""));
  }
  public static void main(String[] args) throws IOException {
    for(int i=0;i<100;i++) {
```

```
        RequestGenerator r=new RequestGenerator();
        String ch1=r.generateAdSlotSize().replace("(","");
        String adSlotSize=ch1.replace(',', ';');
        String ch2=r.generateRegionCity().replace("(","");
        String regionCity=ch2.replace(',', ';');
        String outputFile=r.getClass().
           getResource("BidRequestGenerated.csv").getFile();
        String writer=r.generateBidID()+";"+
           r.generateTimestamp()+";"+
           regionCity.replace(")","")+";"+
           r.generateAdExchange()+";"+
           adSlotSize.replace(")","")+";"+
           r.generateAdSlotVisibility()+
           ";"+r.generateAdSlotFloorPrice()+";"+
           r.generateUserProfileID();
        append(outputFile, writer);
     }
   }
```

Code principal de Publish/Subscribe

```
public void publish(String user_profiles,String device,String
   bidRequest){
     Map<String,Object> map = null;
     BasicProperties props = null;
     try{
       Connection conn = RabbitMQConnection.getConnection();
       if(conn != null){
         Channel channel = conn.createChannel();

         for(String user_profile :user_profiles.split(",")) {
```

```
            props = new BasicProperties();
            map = new HashMap<String,Object>();
            map.put("userProfile",user_profile );
            map.put("device",device);
            props = props.builder().headers(map).build();
            channel.basicPublish(HeaderExchange.EXCHANGE_NAME,
               HeaderExchange.ROUTING_KEY, props,
               bidRequest.getBytes());
            System.out.println(" Message Sent '" +bidRequest + "'");
          }
          channel.close();
          conn.close();
        }
      }catch(Exception e){
        e.printStackTrace();
      }
    }
    public void receive(DSP_pubSub dsp){
      try{
        Connection conn = RabbitMQConnection.getConnection();
        if(conn != null){
          Channel channel = conn.createChannel();
            Consumer consumer = new DefaultConsumer(channel) {
              @Override
              public void handleDelivery(String consumerTag,
                 Envelope envelope, AMQP.BasicProperties properties,
                 byte[] body) throws IOException {
                String message = new String(body, "UTF-8");
                System.out.println(" Message Received '" + message +
                   "'");
                AdX_pubSub.send_bidRequest(dsp, message);
```

```
            }
          };
          channel.basicConsume(dsp.subscription, true, consumer);
        channel.close();
        conn.close();
      }
    }catch(Exception e){
      e.printStackTrace();
    }
  }
```

Code principal du filtrage Top-k basé sur le score

```
public void top_k_scoring(){
  for(DSP_topK dsp: dsps){
        try {
dsp.setScore(l.predictBidPrice(dsp.c1,Long.parseLong(timestamp),
   id_city,adx,ad_height,ad_visibility,floorPrice,user_profiles)
   /kn.predictResponseTime(dsp.t1,Long.parseLong(timestamp),
   id_city,adx,ad_height,ad_visibility,floorPrice,user_profiles));

        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
      }
      Collections.sort(dsps, new SortbyScore());
      participated_dsps=0;
      HeaderExchange ex=new HeaderExchange();
      ex.createExchangeAndQueue();
      for(DSP_topK dsp : dsps){
```

```
            if(participated_dsps<=k) {
              publish(user_profiles, bidRequest);
              receive(dsp);
            }
          }
          auction();
}
  public Classifier process_linear_regression(String trainFile,
     String testFile) throws Exception {

    Instances trainingDataSet = getDataSet(trainFile);
    Instances testingDataSet = getDataSet(testFile);
    Classifier classifier = new
       weka.classifiers.functions.LinearRegression();
    classifier.buildClassifier(trainingDataSet);
    return(classifier);
  }
  public double predictBidPrice(Classifier classifier,long
     timestamp, int cityID, int adExchange, int adSlotHeight,int
     adSlotVisibility, float adSlotBidFloorPrice, int[]
     userProfileIDs) throws Exception {
    Instance inst = new DenseInstance(7);
    inst.setValue(0, timestamp);
    inst.setValue(1, cityID);
    inst.setValue(2, adExchange);
    inst.setValue(3, adSlotHeight);
    inst.setValue(4, adSlotVisibility);
    inst.setValue(5, adSlotBidFloorPrice);
    inst.setValue( 6, userProfileIDs);
    double value = classifier.classifyInstance(inst);
    return value;
```

```
}
public Classifier process_knn(String trainFile, String testFile)
    throws Exception {

  Instances trainingDataSet = getDataSet(trainFile);
  trainingDataSet.setClassIndex(trainingDataSet.numAttributes() -
     1);
  //k - the number of nearest neighbors to use for prediction
  Classifier classifier= new IBk(4);
  classifier.buildClassifier(trainingDataSet);
  return(classifier);
}
public double predictResponseTime(Classifier classifier,long
    timestamp, int cityID, int adExchange, int adSlotHeight,int
    adSlotVisibility, float adSlotBidFloorPrice, int[]
    userProfileIDs) throws Exception {
  ArrayList<Attribute> atts = new ArrayList<Attribute>();
  atts.add(new Attribute("Timestamp"));
  atts.add(new Attribute("CityID"));
  atts.add(new Attribute("AdExchange"));
  atts.add(new Attribute("AdSlotHeight"));
  atts.add(new Attribute("AdSlotVisibility"));
  atts.add(new Attribute("AdSlotBidFloorPrice"));
  atts.add(new Attribute("UserProfileIDs"));
  Instance inst = new DenseInstance(7);
  inst.setValue(0, timestamp);
  inst.setValue(1, cityID);
  inst.setValue(2, adExchange);
  inst.setValue(3, adSlotHeight);
  inst.setValue(4, adSlotVisibility);
  inst.setValue(5, adSlotBidFloorPrice);
```

```
        inst.setValue( 6, userProfileIDs);
        Instances dataUnlabeled = new Instances("TestInstances", atts,
           0);
        dataUnlabeled.add(inst);
        dataUnlabeled.setClassIndex(dataUnlabeled.numAttributes() - 1);
        double value =
           classifier.classifyInstance(dataUnlabeled.firstInstance());
        /** Prediction Output */
        return value;
      }
    public void receive(DSP_topK dsp){
        try{
          Connection conn = RabbitMQConnection.getConnection();
          if(conn != null){
            Channel channel = conn.createChannel();
            Consumer consumer = new DefaultConsumer(channel) {
              @Override
              public void handleDelivery(String consumerTag, Envelope
                 envelope, AMQP.BasicProperties properties, byte[]
                 body) throws IOException {
                participated_dsps++;
                String message = new String(body, "UTF-8");
                dsp.send_bidRequest(message);
              }
            };
            channel.basicConsume(dsp.subscription, true, consumer);
            channel.close();
            conn.close();
          }
        }catch(Exception e){
          e.printStackTrace();
```

```
    }
  }
```

Code principal de Top-k basé sur la sélection binaire

```
public void top_k_classifier(){
   for(DSP_topK_Classifier dsp : dsps){
         if(c.predict(m,d,dsp.name).contentEquals("yes")){
           publish(user_profiles, bidRequest);
           receive(dsp);
         }
  }
  auction();
}
public MLRegression process(VersatileMLDataSet data) {
    ColumnDefinition outputColumn = data.defineSourceColumn("top",
       1,
        ColumnType.nominal);
    data.analyze();
    data.defineSingleOutputOthersInput(outputColumn);
    EncogModel model = new EncogModel(data);
    model.selectMethod(data, MLMethodFactory.TYPE_FEEDFORWARD);
    model.setReport(new ConsoleStatusReportable());
    data.normalize();
    model.holdBackValidation(0.3, true, 1001);
    model.selectTrainingType(data);
    MLRegression bestMethod = (MLRegression)model.crossvalidate(5,
       true);
    return bestMethod;
  }
```

```
public String predict(MLRegression bestMethod,VersatileMLDataSet
   data,String name) {
  NormalizationHelper helper = data.getNormHelper();
  String csv=name;
  String[] line = new String[1];
  MLData input = helper.allocateInputVector();
  line[0] = csv;
  helper.normalizeInputVector(line,input.getData(),false);
  MLData output = bestMethod.compute(input);
  String irisChosen =
     helper.denormalizeOutputVectorToString(output)[0];
  return irisChosen;
}

public void receive(DSP_topK_Classifier dsp){
  try{
    Connection conn = RabbitMQConnection.getConnection();
    if(conn != null){
      Channel channel = conn.createChannel();
      Consumer consumer = new DefaultConsumer(channel) {
        @Override
        public void handleDelivery(String consumerTag, Envelope
           envelope, AMQP.BasicProperties properties, byte[]
           body) throws IOException {
          String message = new String(body, "UTF-8");
          dsp.send_bidRequest(message);
        }
      };
      channel.basicConsume(dsp.subscription, true, consumer);
      channel.close();
      conn.close();
```

```
      }
    }catch(Exception e){
      e.printStackTrace();
    }
  }
```

Code principal de Top-k basé sur le rang

```
public void top_k_rank(){
for(DSP_topK dsp: dsps){
        try {
          dsp.setRank(l.predictRank(c1, dsp.param));

        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
      }
      Collections.sort(dsps, new SortbyRank());
      Collections.reverse(dsps);
      participated_dsps=0;
      HeaderExchange ex=new HeaderExchange();
      ex.createExchangeAndQueue();
      for(DSP_topK dsp : dsps){
        System.out.println(dsp.rank);
        if(participated_dsps<=k) {
          publish(user_profiles, bidRequest);
          receive(dsp);
        }
      }
    auction();
```

```
}
public Classifier process_rank(String trainFile, String testFile)
   throws Exception {

  Instances trainingDataSet = getDataSet(trainFile);
  Instances testingDataSet = getDataSet(testFile);
  Classifier classifier = new
     weka.classifiers.functions.LinearRegression();
  classifier.buildClassifier(trainingDataSet);
  return(classifier);
}
public double predictRank(Classifier classifier,int name) throws
   Exception {
  Instance inst = new DenseInstance(1);
  inst.setValue(0, name);
  double value = classifier.classifyInstance(inst);
  /** Prediction Output */
  return value;
}
public void receive(DSP_topK dsp){
  try{
    Connection conn = RabbitMQConnection.getConnection();
    if(conn != null){
      Channel channel = conn.createChannel();
      Consumer consumer = new DefaultConsumer(channel) {
        @Override
        public void handleDelivery(String consumerTag, Envelope
           envelope, AMQP.BasicProperties properties, byte[]
           body) throws IOException {
          participated_dsps++;
          String message = new String(body, "UTF-8");
```

```
            dsp.send_bidRequest(message);
          }
        };
        channel.basicConsume(dsp.subscription, true, consumer);
        channel.close();
        conn.close();
      }
    }catch(Exception e){
      e.printStackTrace();
    }
  }
```

RÉFÉRENCES

Adya, A., Cooper, G., Myers, D. & Piatek, M. (2011). Thialfi : A Client Notification Service for Internet-Scale Applications. *ACM SIGOPS 23nd Symposium on Operating Systems Principle*.

Antonic, A., Roankovic, K., Marjanovic, M., Pripuic, K. & Arko, I. P. (2014). A mobile crowd-sensing ecosystem enabled by a cloud-based publish/subscribe middleware. *International Conference on Future Internet of Things and Cloud*.

Authors, A. (2018). Scalable Edge Computing Architectures for Low Latency Data Dissemination in Topic-based Publish / Subscribe. *IEEE/ACM Symposium on Edge Computing*.

Barazzutti, R., Felber, P., Fetzer, C., Onica, E., Pineau, J.-f., Pasin, M., Rivière, E. & Weigert, S. (2013). StreamHub : A Massively Parallel Architecture for High-Performance Content-Based Publish / Subscribe Categories and Subject Descriptors. *Distributed Event-based Systems*.

Cañas, C., Pacheco, E., Kemme, B., Kienzle, J. & Jacobsen, H.-A. (2015). GraPS : a graph publish/sybscribe middleware. *16th Annual Middleware Conference*.

Canas, C., Zhang, K., Kemme, B., Kienzle, J. & Jacobsen, H. A. (2017). Self-Evolving Subscriptions for Content-Based Publish/Subscribe Systems. *International Conference on Distributed Computing Systems*.

Cao, F. & Singh, J. P. (2005). MEDYM : Match-early with dynamic multicast for content-based publish-subscribe networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Carzaniga, A., Rosenblum, D. S. & Wolf, A. L. (2001). Design and Evaluation of a Wide-Area Event Notification Service University of Colorado at Boulder. *ACM Transactions on Computer Systems*.

Chen, L., Cong, G., Cao, X. & Tan, K. L. (2015). Temporal Spatial-Keyword Top-k publish/subscribe. *International Conference on Data Engineering*.

Cloudurable. (2017). KAFKA ARCHITECTURE. Repéré à http://cloudurable.com/blog/kafka-architecture/index.html.

Dedousis, D., Zacheilas, N. & Kalogeraki, V. (2018). On the fly load balancing to address hot topics in topic-based pub/sub systems. *International Conference on Distributed Computing Systems*.

Eugster, P. T., Felber, P. A., Guerraoui, R. & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Computing Surveys*.

Farnia, F., Kazerouni, A. & Babveyh, A. (2018). Information-based Feature Selection.

Gascon-Samson, J., Kienzle, J. & Kemme, B. (2017). MultiPub : Latency and Cost-Aware Global-Scale Cloud Publish/Subscribe. *International Conference on Distributed Computing Systems*.

IAB. (2016). *OpenRTB API Specification Version 2.5*. Repéré à https://www.iab.com/guidelines/real-time-bidding-rtb-project/.

Ionescu, V. M. (2015). The analysis of the performance of RabbitMQ and ActiveMQ. *14th RoEduNet International Conference - Networking in Education and Research*.

Jacobsen, H.-A., Pandey, N. K., Vitenberg, R., Zhang, K. & Weiss, S. (2014). Distributed event aggregation for content-based publish/subscribe systems. *8th ACM International Conference on Distributed Event-Based Systems*.

Jauvion, G., Grislain, N., Sielenou, P. D., Garivier, A. & Gerchinovitz, S. (2018). Optimization of a SSP's header bidding strategy using Thompson Sampling. *the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Jayaram, K. R. & Eugster, P. (2010). Parametric Subscriptions for Content-based Publish / Subscribe Networks. *This research is supported, in part, by the National Science Foundation (NSF) under grant 0644013 and 0834529*.

Jayaram, K. R., Jayalath, C. & Eugster, P. (2010). Parametric Subscriptions for Content-Based Publish/Subscribe Networks. *Middleware 2010*.

Ji, S., Ye, C., Wei, J. & Jacobsen, H.-a. Scalable Content-based Publish / Subscribe. *Working Technical Report, initial version : August 2011*.

Ji, S., Ye, C., Wei, J. & Jacobsen, H.-A. (2015). MERC : Match at Edge and Route intra–Cluster for Content-based Publish/Subscribe Systems. *Middleware '15*.

Juniper. (2016). AI MACHINE LEARNING TO DRIVE 'REAL TIME BID' ADVERTISING SPEND TO $42BN GLOBALLY BY 2021. Repéré à https://www.juniperresearch.com/press/press-releases/ai-machine-learning-drive-real-time-bid-ad-spend.

Kalra, A., Borcea, C., Wang, C. & Chen, Y. (2019). Reserve price failure rate prediction with header bidding in display advertising. *the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

KAUSHIK, S. Introduction to Feature Selection methods with an example (or how to select the right variables ?). Repéré à https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/.

Kumar, J. (2017). Timeout Analysis, Troubleshooting and Notification in Real Time Bidding Advertising System with Implementation. *Computer Science and Engineering*.

Lee, K. C., Jalali, A. & Dasdan, A. (2013). Real time bid optimization with smooth budget delivery in online advertising. *the 7th International Workshop on Data Mining for Online Advertising, ADKDD 2013 - Held in Conjunction with SIGKDD 2013*.

Liao, H., Peng, L., Liu, Z. & Shen, X. (2014). iPinYou Global RTB Bidding Algorithm Competition Dataset. *20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Mullarkey, M. T. & Hevner, A. R. (2015). New Horizons in Design Science : Broadening the Research Agenda. *Desrist 2015*. Repéré à http://www.scopus.com/inward/record.url?eid=2-s2.0-84937485125{&}partnerID=tZOtx3y1.

Nzengang, F. V. (2009). Introduction to Mixed integer nonlinear programming.

Oppen, D. C. (1980). Complexity, convexity and combinations of theories. *Theoretical Computer Science*.

Pandey, N. K., Zhang, K., Weiss, S., Jacobsen, H. A. & Vitenberg, R. (2015). Minimizing the Communication Cost of Aggregation in Publish/Subscribe Systems. *International Conference on Distributed Computing Systems*.

Pietzuch, P. R. & Bacon, J. M. (2002). Hermes : A distributed event-based middleware architecture. *International Conference on Distributed Computing Systems*.

Qin, R., Yuan, Y. & Wang, F. Y. (2017). Optimizing the revenue for ad exchanges in header bidding advertising markets. *2017 IEEE International Conference on Systems, Man, and Cybernetics*.

RabbitMQ. (2019). Part 4 : RabbitMQ Exchanges, routing keys and bindings. Repéré à https://www.cloudamqp.com/blog/2015-09-03-part4-rabbitmq-for-beginners-exchanges-routing-keys-bindings.html.

Ratnasamy, S., Handley, M., Karp, R. & Shenker, S. (2001). Networked Group Communication. *Networked Group Communication*.

Romero, C. & Oliveira, H. P. (1989). Exact solutions in brans-dicke theory : A dynamical system approach. *Astrophysics and Space Science*.

Rowstron, A., Kermarrec, A. M., Castro, M. & Druschel, P. (2001). Scribe : The design of a large-scale event notification infrastructure. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Salehi, P., Zhang, K. & Jacobsen, H.-A. (2017). PopSub : Improving Resource Utilization in Distributed Content-based Publish/Subscribe Systems. *11th ACM International Conference on Distributed and Event-based Systems*.

Sayedi, A. (2018). Real-time bidding in online display advertising. *Marketing Science*.

Sharma, Y., Ajoux, P., Ang, P., Callies, D., Choudhary, A., Demailly, L., Fersch, T., Guz, L. A., Kotulski, A., Kulkarni, S., Kumar, S., Li, H., Li, J., Makeev, E., Prakasam, K., Renesse, R. V., Roy, S., Seth, P., Song, Y. J., Wester, B., Veeraraghavan, K., Xie, P., Nsdi, I., Sharma, Y., Ajoux, P., Ang, P., Callies, D., Choudhary, A., Kulkarni, S., Kumar, S., Li, H., Li, J., Makeev, E., Prakasam, K., Renesse, R. V., Roy, S. & Seth, P. (2015). Wormhole : Reliable Pub-Sub to Support Geo-replicated Internet Services. *Usenix Nsdi*.

Shraer, A., Gurevich, M., Fontoura, M. & Josifovski, V. (2014). Top-k publish-subscribe for social annotation of news. *the VLDB Endowment*.

Thein, K. M. M. (2014). Apache Kafka : Next Generation Distributed Messaging System. *2014 IJSETR*.

Van Steen, M., Urdaneta, G., Setty, V., Vitenberg, R., Gimåker, S. & Kreitz, G. The hidden pub/sub of spotify. *7th ACM international conference on Distributed event-based systems*.

Wang, J., Zhang, W. & Yuan, S. (2016). Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting.

Wu, W. C. H., Yeh, M. Y. & Chen, M. S. (2015). Predicting winning price in real time bidding with censored data. *the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yang, L., Hajiesmaili, M. H., Sitaraman UMass Amherst, R., Mallada, E., Wong, W. S., Wierman, A. & Sitaraman, R. (2019). Online Inventory Management with Application to Energy Procurement in Data Centers. *ACM Reference Format*.

Yuan, Y., Wang, F., Li, J. & Qin, R. (2014). A survey on real time bidding advertising. *IEEE International Conference on Service Operations and Logistics, and Informatics*.

Zhang, C. R. & Zhang, E. (2014). Optimized bidding algorithm of real time bidding in online ads auction. *International Conference on Management Science and Engineering*.

Zhang, K., Sadoghi, M., Muthusamy, V. & Jacobsen, H.-A. Efficient covering for top-k filtering in content-based publish/subscribe systems. *Middleware '17*.

Zhang, K., Muthusamy, V. & Jacobsen, H. A. (2012). Total order in content-based publish/subscribe systems. *International Conference on Distributed Computing Systems*.

Zhang, K., Sadoghi, M., Muthusamy, V. & Jacobsen, H. A. (2013). Distributed ranked data dissemination in social networks. *International Conference on Distributed Computing Systems*.

Zhang, W. & Wang, J. (2015). Statistical Arbitrage Mining for Display Advertising. *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Zhang, W., Yuan, S. & Wang, J. Optimal real-time bidding for display advertising. *20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.